Final Degree Research Thesis

Fast Video Object Segmentation by Pixel-Wise Feature Comparison

A Real-Time approach to Video Object Segmentation

Degree in Industrial Engineering
&
Degree in Telecommunications Engineering

Author:
Rafel Palliser Sans

Supervisor:
Sanja Fidler

Home Tutor:
Xavier Giró i Nieto

May 2019

Rafel Palliser Sans: *Fast Video Object Segmentation by Pixel-Wise Feature Comparison. A Real-Time approach to Video Object Segmentation*©, May 2019.

A Final Degree Research Thesis submitted to the Escola Tècnica Superior d'Enginyeria Industrial de Barcelona (ETSEIB), Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona (ETSETB) and Centre de Formació Interdisciplinària Superior (CFIS) of the Universitat Politècnica de Catalunya (UPC) in partial fulfillment of the requirements for the Doble Titulació en grau d'Enginyeria Industrial i grau d'Enginyeria en Telecomunicacions, menció Telemàtica.

Thesis produced at the Vector Institute, University of Toronto, under the supervision of Prof. Sanja Fidler.

Supervisor:
Sanja Fidler

Home Tutor:
Xavier Giró i Nieto

Location:
Toronto, ON, Canada

If we want machines to think, we need to teach them to see.

— Fei-Fei Li

Abstract

Fast Video Object Segmentation by Pixel-Wise Feature Comparison is a Real-Time approach to Video Object Segmentation, which tackles the task of segmenting a video sequence with multiple objects with pixel resolution: in every frame, each pixel must belong to an object. Solving Video Object Segmentation would help a machine recognizing real-world scenes by detecting and tracking objects. However, it can also be used as a tool to build better and larger datasets easily, tagging all this ridiculous amount of videos that are uploaded to the internet every hour.

The method that we present in this thesis is trained on the well-known DAVIS dataset and evaluated on its own benchmark, taking into account not only its accuracy but also its speed, aiming to be used in Real-Time applications. The final version of Fast Video Object Segmentation by Pixel-Wise Feature Comparison (whose code is published at github.com/rafelps) achieves a $\mathcal{G}$ Mean score of 71.1% on the DAVIS 2017 validation set while running at 18 FPS, state-of-the-art results in both the accuracy and the processing speed.

This thesis contains a concise introduction to Deep Learning and Computer Vision as well as important Video Object Segmentation previous work, and a full explanation of our approach to the task presenting ablation studies on our different versions and modules, and a comparison between our method and DAVIS benchmark's best solutions.

KeyWords: Artificial Intelligence · Machine Learning · Deep Learning · Convolutional Neural Networks · Computer Vision · Video Object Segmentation · DAVIS dataset · ResNet · Feature Extraction · Pixel Embeddings · Pixel-Wise · Real-Time · PyTorch.

Acknowledgments

First of all, I would like to show my greatest gratitude to my supervisor Prof. Sanja Fidler, for accepting me in her research group and guiding me in my investigation project presented in this thesis.

I would also like to thank all Sanja's group, and specially Xiaohui Zeng for helping and giving me valuable advice. It has been a pleasure to be part of that team, which is wondrous both personally and professionally.

I also need to thank the Vector Institute for its marvelous staff, its lively workspace, and its research community. Thank you for letting me use your offices, your computing resources and for all the high-quality talks and activities you organized. I could not imagine a better place to work on my project.

I would also like to thank the CFIS, Prof. Miguel Ángel Barja, Prof. Eduardo Alarcón and Prof. Toni Pascual for the opportunities given to me. Being able to produce my final degree research thesis in such an outstanding place as Vector Institute is, and being supervised by one of the most leading AI researchers worldwide, would not have been possible without your work. Thanks also to Prof. Xavier Giró for being my home tutor, and to CFIS' private sponsors for the grants given to the CFIS international mobility program students.

I also wish to thank Eric Guisado, Ariadna Matabosch, Didac Surís and Jordi Fortuny for helping me when things were not going as expected and for celebrating together when good results arrived. Outside the office, they were my family abroad, thanks for sharing many good moments.

Finally, I would like to thank my parents Rafel and Brigit, Miquel, and Fàtima for encouraging me pursuing this opportunity and sending me the warmth of my home, Menorca, in the distance.

Contents

List of Figures

List of Tables

Glossary

AI:	Artificial Intelligence
BCE:	Binary Cross-Entropy Loss
CNN:	Convolutional Neural Network
CV:	Computer Vision
DL:	Deel Learning
DNN:	Deep Neural Network
ML:	Machine Learning
NN:	Neural Network
RGB:	Red, Green and Blue. Used to refer to color images
RNN:	Recursive Neural Network
VOS:	Video Object Segmentation

1. Introduction

1.1 Motivation

Artificial Intelligence is one of the most important technologies today. A proof of it can be seen, for example, in the automobile industry, where a few years ago electronics were the most significant innovations while nowadays everybody is talking about autonomous driving and other AI driving assistance. Moreover, some years ago, doctors struggled to diagnose some diseases; today machines help them improving their exactness in the diagnosis by recognizing tumors or analyzing electroencephalograms.

Artificial Intelligence is growing by leaps and bounds, and it is inevitable that we share our lives with AI machines in the next years. For this reason, it is an enormous motivation for me to carry on my final degree research thesis in this field.

Furthermore, as an engineer, I like to work on projects with clear applications in the real world, and Video Object Segmentation is an obvious example to it. Firstly, it processes videos which contain the temporal factor that image processing lacks. Then, it detects all types of objects what has many immediate applications like collision avoidance or helping robots interacting with real environments. Finally, it is a segmentation problem, which means that the detections are made in the highest resolution possible, pixel-wise.

1.2 Objectives

The first goal of this final degree research thesis is to understand how AI works without any prior knowledge of the technology. It is impossible to achieve good results without comprehending the fundaments and basics of its operation.

On the same direction, it is also crucial to get in touch with the previous work done in the specific task of Video Object Segmentation. Understanding the differences between the released methods, the reason of their architectures, how and why they work, and which are their weaknesses is a good starting point to begin designing our solution.

In terms of our model, there are several objectives that we want to accomplish. The most important one is that our method has to be fast. Fast enough to be used in Real-Time applications. As a consequence, an implicit objective is that our model is simple. On the other side, another essential aim is that it tackles the problem of multi-object segmentation in an intention to face real-world environments.

Last but not least, our designed architecture and method has to be implemented to validate its accuracy, speed, and effectiveness, what involves learning a new programming language or a Machine Learning framework to produce an efficient and clean implementation.

2. Artificial Intelligence, Deep Learning and Computer Vision

This chapter contains a brief introduction to Artificial Intelligence, and more specifically to the Deep Learning and Computer Vision technologies, jointly with a presentation of the most basic Neural Networks, and the main procedure to train them. In short, this chapter includes all the relevant prior knowledge needed for the research project presented in this thesis. It is the entrance door to the AI world.

2.1 Artificial Intelligence

Artificial Intelligence (AI) is used in many fields today such as data prediction, image classification, object detection or face recognition. It is an already immense but still growing world, and thus, one can find many definitions of it. Even so, the following one may contain all the necessary ingredients to describe it:

> "Artificial intelligence is defined as the branch of science and technology that is concerned with the study of software and hardware to provide machines the ability to learn insights from data and the environment, and the ability to adapt to changing situations with high precision, accuracy and speed."
>
> — Amit Ray, Compassionate Artificial Superintelligence AI 5.0

So one could wonder: how can machines be provided the ability to learn? As Prof. Geoffrey Hinton once said, "the only way to get artificial intelligence to work is to do the computation in a way similar to the human brain". The human brain is what gives us intelligence, what makes the human race different among all the other species. Although we still do not know exactly how our brain works, we know that it has a neural structure, so the AI aspiration is to replicate this structure into machines.

However, not only the architecture is replicated but also the way that humans learn. No child knows how to ride a bike from the first day he tries. In fact, it happens just the opposite, he slowly learns how to do it by practicing. The AI learning methodology is based on a repetitive approach, where the model gradually acquires knowledge by iterating.

2.2 Deep Learning

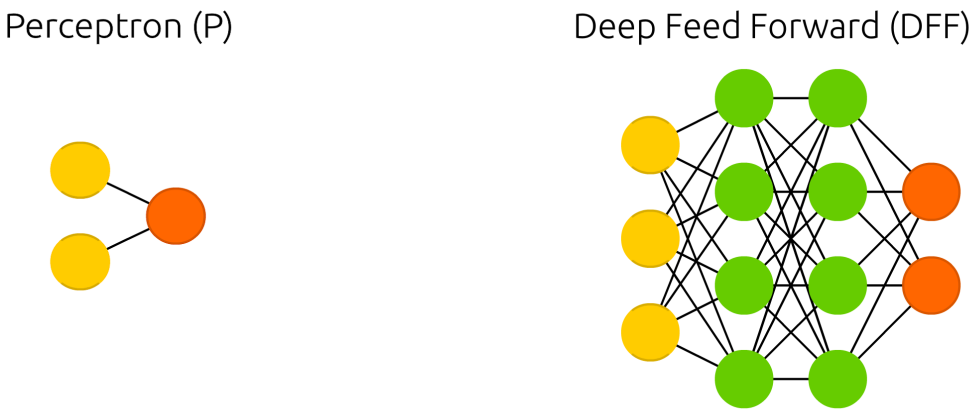
2.2.1 Neural Networks

A Neural Network (NN) is, as its name says, a group of connected neurons. Biologically, a neuron has multiple stimuli (inputs) and generates a specific output signal depending on their combination. By simulating the human brain, in the AI world, a neuron is a node that performs the following linear function between its inputs $\mathbf{x}$, and its output signal y:

$$y = \mathbf{W}^\top \mathbf{x} + \mathbf{b} \tag{2.1}$$

where $\mathbf{W}$ and $\mathbf{b}$ are the neuron's weights and bias respectively, two own parameters.

When we start grouping neurons such that the outputs of one layer are used as inputs for the succeeding one, we talk about a network. Every network has at least an input layer and an output one. However, any number of layers can be added between them. These are called hidden layers, and a NN with more than one of them is called Deep Neural Network (DNN). In the Figure 2.1a we can see the simplest network possible: a single output neuron with two inputs. On the other side, in Figure 2.1b, we can see an example of a DNN. Other configurations can be seen in the Network Zoo, in Appendix A.



(a) A neuron with two input cells.

(b) A DNN with 3 input cells, 2 hidden layers with 4 neurons each and 2 output cells.

Figure 2.1: Simple Neural Networks. Input cells shown in yellow, hidden layers in green and output cells in orange.

2.2.2 Training Neural Networks

Training a NN means to reach a condition where the network has understood the data, and thus, it is able to make accurate predictions on it. There are many different network architectures, and data types, still, there are some generic steps that are used in most of the training schedules.

In the first place, our data may need a preprocessing step depending on the dataset and chosen network, as many of them require normalized data. Furthermore, some architectures only accept input images with specific dimensions, so these may have to be resized or cropped. It is crucial that the data matches the network's requirements.

Once the data is in the correct format, it is time to forward-pass a data point into the network. As each neuron has a linear function (see Equation 2.1), we need to introduce some non-linearities. Otherwise, the composition of various linear functions would become a single linear function, and we would not take advantage of the hidden layers we introduced. These non-linear functions are called activation functions and are placed at the output of almost each neuron. ReLU (Figure 2.2) and Sigmoid (Figure 2.3) are the most used ones. While the former converts all negative outputs to zero, the later smoothly transforms the outputs from $\mathbb{R}$ to the range $[0, 1]$:

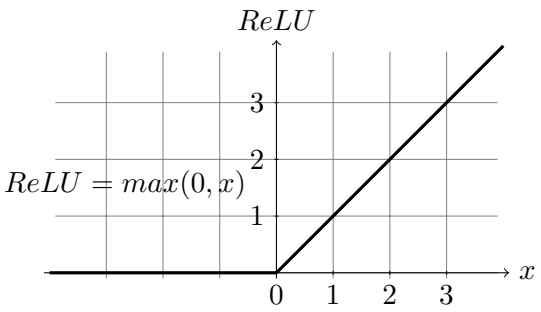


Figure 2.2: Rectified Linear Unit (ReLU).

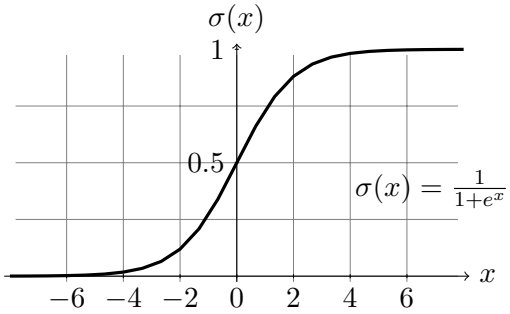


Figure 2.3: Sigmoid function.

When the data point has forward-passed all the neurons and activation functions and arrives at the output of the last cell, a normalizing function may be used instead of an activation one. For example, in case that we have two outputs that should be probabilities, we would use a SoftMax function to normalize the outputs and force them sum to one. For a vector $\mathbf{z}$ of dimension K, the normalized vector produced by SoftMax function $\sigma(\mathbf{z})$ is:

$$\sigma : \mathbb{R}^K \rightarrow \ [0,1]^K$$

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=0}^{K-1} e^{z_k}} \qquad \text{for all } j \in \{0, 1, \ldots, K-1\} \tag{2.2}$$

After this function, our data point has been forward-passed through the whole network. To train our model, however, we need to give it some feedback on how accurate the predictions were. This accuracy is calculated in the loss or cost functions. In unsupervised methods, the loss functions compare different predictions between them, while when training in a supervised way, the predictions are compared to the ground truth. Some of the most used cost functions in supervised methods are L2 distance and Cross-Entropy Loss. The former is used in regression problems, where the target is continuous (see Equation 2.3 where $\hat{\boldsymbol{y}}$ is the prediction and $\boldsymbol{y}$ is the ground truth). On the other hand, Binary Cross-Entropy Loss is utilized in classification problems where the target is discrete (see Equation 2.4, where y denotes the ground truth, and p is the probability for the predicted value of belonging to class 1).

$$\mathcal{L}_{L2}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sqrt{\sum_{k=0}^{K-1} (y_k - \hat{y_k})^2} \qquad \boldsymbol{y} = (y_0, y_1, \ldots, y_{K-1}), \quad \hat{\boldsymbol{y}} = (\hat{y}_0, \hat{y}_1, \ldots, \hat{y}_{K-1}) \tag{2.3}$$

$$\mathcal{L}_{BCE} = -y \log(p) + (y-1) \log(1-p) \qquad y \in \{0, 1\}, \quad p \in [0, 1] \tag{2.4}$$

Once we have calculated the cost or error, we need to update the parameters (weights) of the neurons, so they adapt to the specific problem or data we are trying to learn. There are many optimizing algorithms such as Stochastic Gradient Descent (SGD), SGD Momentum, Adam, ... All of them are first-order algorithms, which means that they need the first order derivative of what we are minimizing (the cost) by respect to the parameters we want to optimize (the weights). There are higher-order algorithms, but they are not that popular.

The Backpropagation algorithm gives us these first order derivatives by differentiating the loss by respect to each weight by applying the chain rule inversely, backward. Note that this algorithm needs all the forward pass functions (the neuron's forward function, activation functions, normalizing functions and the cost function itself) to be differentiable. It has to be remarked that one of the most used activation functions, ReLU, is not differentiable at 0. However, an approximation of its first derivative is employed, so there is no trouble in using it.

These are the essential steps to train a network and have to be performed iteratively with all the data points used in training. In addition to this, there are a couple of other important concepts. On the one hand, if we use more than one data point in each forward-backward pass, we call it a batch. A larger batch size gives the optimizer more stability but needs more memory. On the other hand, when the whole training data has been passed through the network, we call it an epoch. It usually takes several epochs to train a network from scratch, although it is problem-dependent.

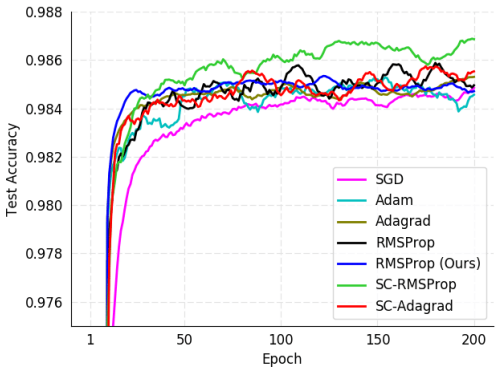


Figure 2.4: Accuracy curve when training MNIST using different optimizers.

2.3 Computer Vision

Many different problems are being solved using Deep Learning today. Thus, one can find numerous different network structures, each of them optimized to be used for a specific task, or with a particular data type.

Machine Learning techniques and simpler NN are used for Big Data Analysis, for example. Problems such as Natural Language Processing (NLP) that rely on sequences (a series of words, for example), use Recursive Neural Networks (RNN) because its architecture has some memory to take advantage of previous outputted words. Additionally, there are different methods to train NN's. Reinforcement Learning (RL) techniques rely on a reward function that they try to maximize while Generative Adversarial Networks (GANs) are two-role structures where the generator learns how to produce new synthetic data, and the discriminator learns how to distinguish synthetic from real data. Once trained, the discriminator cannot differentiate synthetic instances and so the generator, who has understood the dataset, can be used to create new data points.

One could write an entire book talking about different Neural Network structures. Nevertheless, this thesis is focused on Computer Vision and Convolutional Neural Networks.

Computer Vision (CV) is a field of AI that aims at giving machines a visual understanding of images and videos similar to how human vision works. Its main difference with other techniques and architectures remains in its data type. Images have a clear two-dimensional spatiality, in

contrast to other kinds of data such as text, audio or simple scalars. In addition to this, a single image contains much more information than other data. For example, we could think of a Big Data problem where each data point can have thousands of features. A single RGB image at low resolution (480p) has 3 channels $\times$ 480 $\times$ 854 = 1229760 values, three orders of magnitude larger, which encourages the idea that different architectures are needed.

In fact, these specific architectures utilized in Computer Vision, are called Convolutional Neural Networks and are used in many different tasks:

- **Image Classification:** This is one of the simplest tasks. Each image in the dataset contains a single object, which belongs to a previously known set of categories. The intention is to predict the object class.
- **Localization + Classification:** In this task, in addition to predicting the object category, the network has to prognosticate the object position inside the image by printing a bounding box around it.
- **Object detection:** In this problem, each image can contain multiple objects, and the goal is to detect them by printing a bounding box around every object in the picture. It can include object classification (among a set of known classes) or be class agnostic.
- **Semantic Segmentation:** To semantically segment an image means to assign each of the pixels to one of the possible classes, with prior knowledge of the collection of appearing categories, which usually includes a *background* class.
- **Instance Segmentation:** Adds the difficulty of distinguishing instances of the same class. When assigning a label to each pixel in an image that contains, for example, two people, pixels belonging to the first person should be classified as *personA*, while pixels belonging to the other should be labeled as *personB*.

All the above are image processing tasks, but can be easily extrapolated to video sequences, as a video is no more than a sequence of frames. Some examples could be classifying a video where a single object appears, semantically segmenting all the pixels in every frame of the video or detecting and tracking a target throughout the whole sequence.

2.3.1 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a type of NN's specifically designed to process images as they use their spatiality to extract additional information, and have a reduced number of weights in comparison to standard Neural Networks. CNN architectures are composed of three types of layers: convolutional layers, which are the most important and give the name to the whole network, pooling layers, and fully connected layers.

Convolutional layers are made of convolutional filters, and each of these filters also called kernels, has some learnable weights that are combined with the images' values by dot products and sums. Figure 2.5 shows an example of a convolutional layer, with a 3x3 filter (9 learnable weights). The center of the kernel is placed above a pixel of the image, called source pixel, and it is weighted, jointly with its neighbors, by the kernel values. Finally, the results of all the products are added to produce one output value. The filter is then shifted to every other position to produce all the other outputs.

Furthermore, convolutional layers have some essential factors that slightly modify their behavior. On the one hand, we have the padding factor. To be able to operate in the edges, zero-padding or mirror-padding can be performed. On the other hand, the stride factor indicates

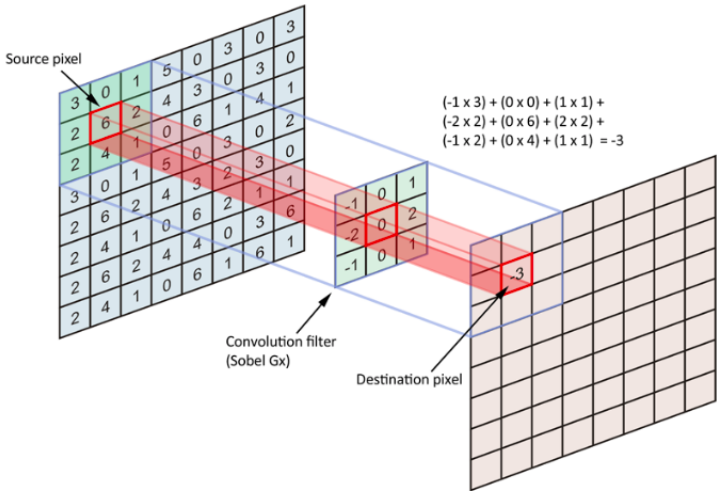


Figure 2.5: A convolutional layer with a single 3x3 filter.

how many positions the kernel has to be shifted between operations. By default, stride one is used, so the center of the filter is placed to every input pixel, and spatial dimensions are conserved. However, larger strides can be employed to reduce output dimensionality as the center of the kernel is only placed to fewer input pixels.

Each of these convolutional filters produces a two-dimension output, which we call a feature map. However, in many computer vision tasks, more features drive to better predictions, so convolutional layers often have multiple filters, usually from 64 to 2048, each of them with its learnable parameters. For example, let's think about a convolutional layer with 128 7x7 filters. If the layer's input is an RGB image, each filter results in having $7 \times 7 \times 3 = 147$ learnable weights, as color images have 3 channels. As we have 128 of them, we end up with approximately 19000 parameters and 128 feature maps, in front of the single output generated by a non-convolutional layer, with extremely more parameters (1230000 for a 480p resolution image). This is the potential of CNN's and why they are perfect for image processing: they work with two dimensions (or more), incorporate context information in their operations, and save tons of computation resources.

In addition to convolutional layers, almost any CNN architecture incorporates some pooling layers. These are layers without learnable weights, so they are free in terms of computing memory, and its job is to reduce the spatial dimension of processed images. The idea is that extracted features at original dimension can give local information such as edges or colors but once the image is reduced using a pooling layer, resulting features have more global information, which is very useful for image classification, for example.

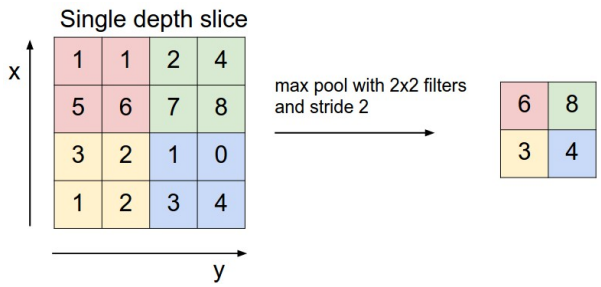


Figure 2.6: A 2x2 Max Pooling layer.

The most known pooling layers are Max Pooling and Average Pooling, and its behavior is very intuitive. Figure 2.6 shows a 4x4 input image passed through a 2x2 Max Pooling layer. Firstly, the image is split into regions of the layer's dimension (2x2 in the example). Afterward, a reduction function is applied to each area generating an output value. In this case, the max operation is used while in Average Pooling, the output value would be calculated by the mean of the input region's values.

Finally, there is a third kind of layer called fully connected layer (FC). These layers are usually used at the end of the networks, to sum up, and get the final predictions. For example, if we are classifying cats and dogs, the last layer would be a two cell FC layer, where each cell would give the probability of that image to contain one or the other animal. Each of these cells is connected to all of the outputs of the previous layer, so they are computationally costly because they have lots of learnable parameters. For example, classifying cats and dogs from an extracted feature space of 512 7x7 feature maps would imply $7 \times 7 \times 512 = 25088$ weights for each cell, resulting in more than 50000 parameters for a single FC layer. One way of controlling it is by using some pooling layers before it, so the dimensions become smaller, and so does the number of learnable weights.

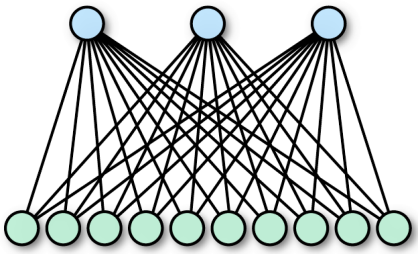


Figure 2.7: A Fully Connected layer with 3 output cells.

2.3.1.1 Most used CNN architectures

Although there are infinite combinations of layers, some of them work better than others in many tasks. The most used architectures, present in more than 85% of today networks are VGG [27] and ResNet [9]. Both of them were well-placed networks of the ImageNet Large Scale Visual Recognition Competition (ILSVRC) [26], an image classification challenge using the ImageNet dataset [5], which contains more than 14 millions annotated images belonging to more than 20000 different categories. The ILSVRC is, in fact, an excellent example to show the Computer Vision history.

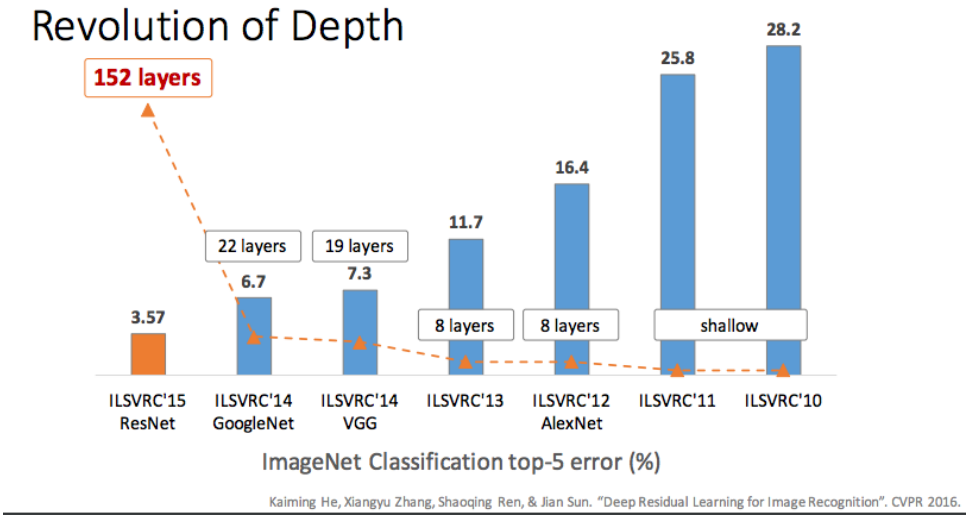


Figure 2.8: ILSVRC. Evolution of top-5 error and architecture depth.

On 2012, Alex Krizhevsky supervised by Geoffrey Hinton won the competition with AlexNet [15]. It was the first CNN that won the challenge and made a turning point as reduced the error rate by 10% to previous non-convolutional methods. The following year the first classified reduced the error by 5%, but the next revolutionary network was the 2014 runner-up, VGG [27], with more than three times the number of parameters, and more than twice the amount of layers, being able to reduce the top 5 error rate to 7.3%. They proved that it is better to have more layers with smaller kernels than vice versa, resulting in the same receptive field. Although it was the runner-up, it achieved the best top 1 error percentage (the benchmark is top 5 error).

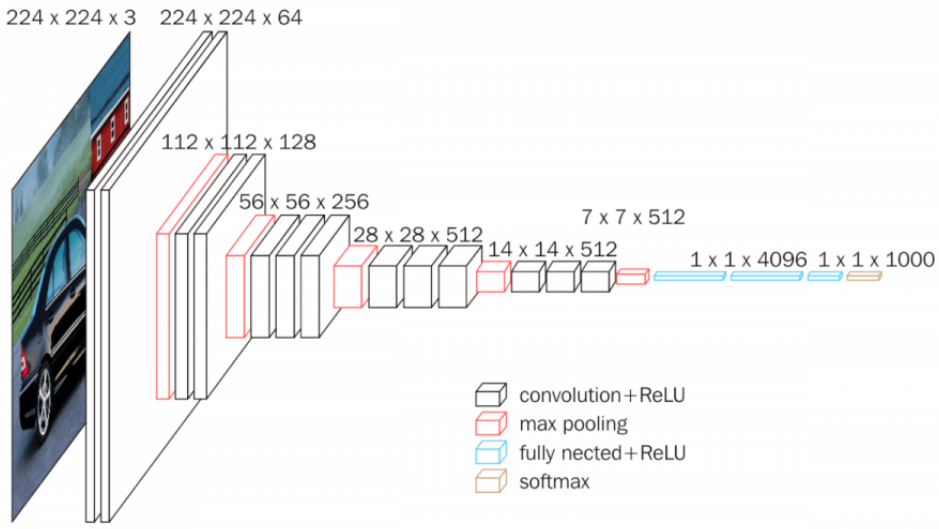


Figure 2.9: VGG's architecture.

Also on 2014, GoogLeNet [28] (playing tribute to Yan LeCun's LeNet [16]), won the competition by increasing the number of layers to 22 but reducing the number of parameters significantly. They managed it by avoiding using FC layers. Moreover, they proposed some new ideas to train the network more efficiently, as introducing auxiliary outputs so the backward gradients could arrive to the first layers in better conditions.

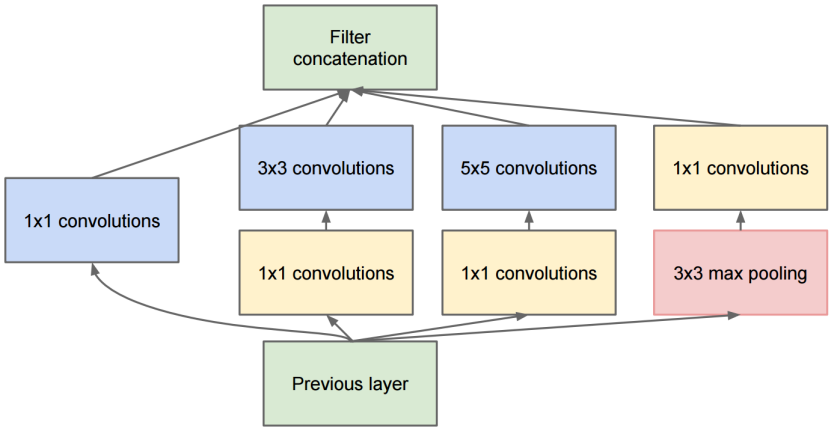


Figure 2.10: GoogLeNet's inception module.

Finally, in 2015, the revolution of depth arrived. ResNet [9], the winner of that year's competition, is a 152-layer residual network and achieved better accuracy than humans, which usually make a 5% error rate in the same task.

ResNet architecture is made of residual blocks. While a regular convolutional layer learns how to turn the input into the correct output, ResNet blocks only calculate which increment has to be added to the input identity to achieve the accurate output. These blocks have excellent properties such as better back-propagation and easiness in training the whole network, which is very important due to the immensity of itself and difficulty of gradients to not be vanished or exploded.

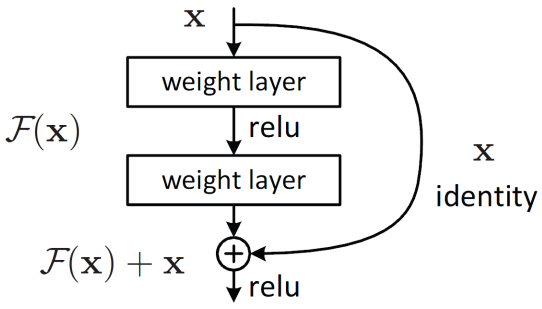


Figure 2.11: ResNet's residual block.

The project presented in this thesis uses a ResNet-50 backbone, as it has a perfect combination of limited use of computational resources, great extracted features and fast execution time.

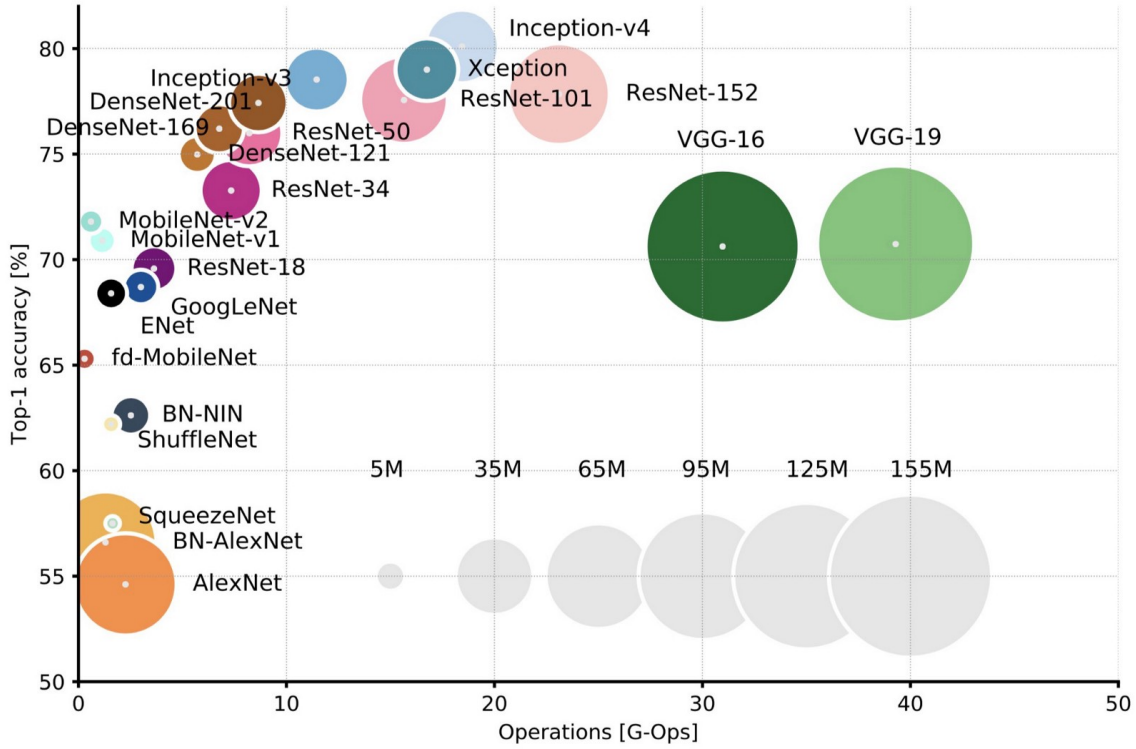


Figure 2.12: Comparison of the number of parameters, computational cost, and accuracy of the most known architectures.

3. Video Object Segmentation

This chapter includes the most relevant information about the task that the project presented in this thesis is treating: Video Object Segmentation. It includes a concise introduction to VOS, the existing VOS datasets, its most important benchmark to compare existing methods, and the most interesting approaches that have been proposed to solve the problem.

3.1 Video Object Segmentation: The task

Video Object Segmentation (VOS) tackles the problem of assigning a label to each pixel in each frame of a video shot, where a shot is defined as a sequence of frames without camera cuts. There are two main variants of the problem regarding the objects to be segmented: in the foreground-background task, a single object has to be separated from its background, while in the multi-object task, several objects have to be differentiated between them, even if belonging to the same category. This thesis is focused on multi-object VOS because it is a task closer to real-world environments, where different objects coexist and occlude each other. Furthermore, foreground-background can be seen as a specific case of multi-object where all the object masks are merged.

Moreover, VOS also has its variants in terms of supervision. Firstly, there is an interactive approach which aims to semi-automatically segment videos with the supervision of a human-in-the-loop, which can add control points to help the network know which objects has to segment. Secondly, we find the unsupervised or zero-shot approach, where the network has to decide which objects have to be segmented without any supervision. In this case, the network's input are only the RGB images of the video sequence. Finally, there is the one-shot VOS, which is the most known and relevant approach today. It is called one-shot because the ground truth masks for the first frame are given, so we have one reference to identify which objects have to be segmented and how they look. This variant is also called weakly-supervised VOS or semi-supervised VOS. Even with these names, they are trained using supervised methods.

Furthermore, it is also important to know that Video Object Segmentation is a class agnostic task, which means that different objects that have to be segmented have to be given a unique label, but they do not have to be categorized into one class. However, some datasets have semantic annotations for further supervision.

Figure 3.1 shows a multi-object class-agnostic one-shot VOS sequence, where the RGB frames for all the video and the ground truth masks for the first frame are given. Note that the masks are pixel-wise annotated, which means that each pixel has a label, which in this case is a color (the black color is reserved for the background). The objective is to extract the maximum information from the first frame's masks to be able to predict where the objects are in the following frames, shadowed in grey. The fact that the green object is a soccer ball it is not important. However, it is crucial that all the pixels belonging to the soccer ball have the same tag in all the predicted masks.

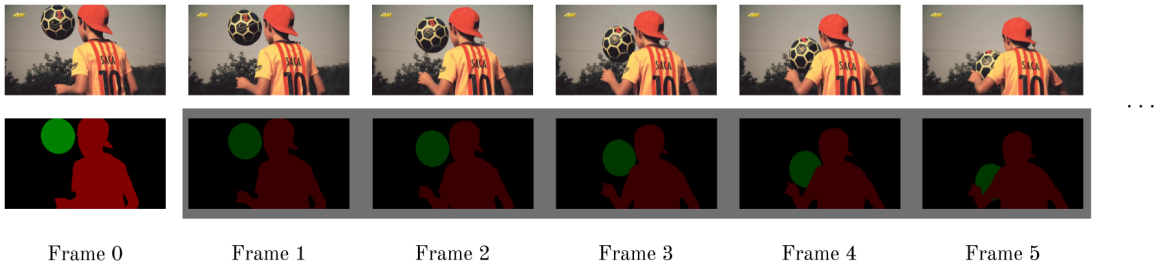


Figure 3.1: One-shot VOS with two objects to segment from the background: Video frames and first frame ground truth masks are given. Shadowed, the goal, which is to predict all the other masks.

3.2 Video Object Segmentation datasets

Video Object Segmentation is a relatively new task, and therefore there are few datasets. Annotating segmentation masks for videos is a tedious and tough work, which also makes it difficult to have big datasets like ImageNet [5], CIFAR10 [14] or MNIST [17], that have image classification annotations.

SegTrack v2 [18] was in 2013 one of the first VOS datasets that appeared. It is formed by only 14 video sequences where 8 of them have a single object, 6 sequences contain 2 objects, and the penguin video exceptionally includes 6 instances. The sequences have an average of 70 frames although some of them have just 20, and others have more than 250. It is a tiny and varied dataset, which makes it challenging to train on it.

Later in 2016, DAVIS [23] dataset was presented, accompanied by an official benchmark. On 2017, a new multi-object version was released, jointly with a VOS challenge that already has 3 editions: 2017, 2018 and 2019. All these releases have made DAVIS the state-of-the-art dataset and benchmark for VOS, as different methods can be easily compared.

As DAVIS is not an enormous dataset, synthetic data or other non-VOS datasets are often used to train VOS methods more favorably before they are fine-tuned on the definitive data. YouTube-Objects [13], for example, has 720000 weakly annotated frames in terms of bounding boxes instead of segmentation masks, for the 10 categories of Pascal VOC Challenge [7]. Furthermore, Pascal VOC 2012 [6] extends its former challenge set of classes from 10 to 20 and provides object segmentation masks for almost 7000 static images. Finally, COCO [20] provides instance level segmentations for more than 90 object classes with an average of 10000 static images per class.

Recently, YouTube-VOS [33] has been released together with its own benchmark. It is almost 50 times larger than DAVIS in terms of frames and object variety, so it will become as known and as important as DAVIS sooner or later.

3.2.1 The DAVIS dataset and benchmark

DAVIS is the state-of-the-art benchmarking dataset for Video Object Segmentation. Its first version, DAVIS 2016 [23], contains 30 training videos and 20 validation sequences. The segmentations provided are foreground-background style, which means having a single mask per frame, even if there is more than one object.

The following year, DAVIS 2017 [24] was released. It extends DAVIS 2016 training set to 60 videos, and its validation set to 30. Furthermore, the annotations are changed to be used in a multi-object instance segmentation environment. The same year DAVIS Challenge [3, 24] was presented with two new sets of videos: test-dev set, with 30 new sequences and another 30 videos exclusively used in the challenge.

In DAVIS 2017 there are approximately 10500 annotated frames, leading to an average of 70 frames per sequence, and a mean of 2.5 objects per video. Although the dataset is not immense, their videos contain a wide variety of objects, huge pose variations, scale changes and occlusions and reappearances between the instances. These characteristics make the dataset incredibly challenging, and the proof of it is that the best method today has an accuracy lower than 70% on it while having more than 80% precision in other datasets.

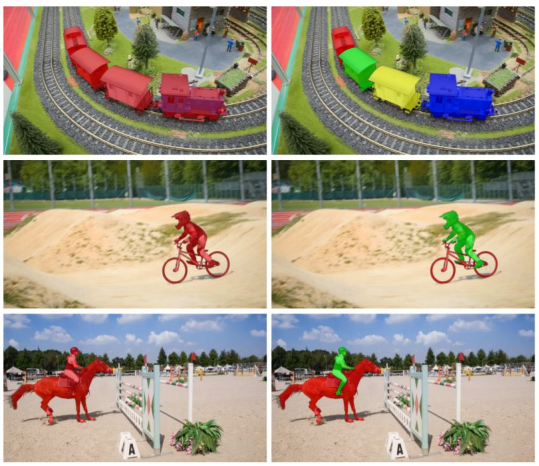


Figure 3.2: Comparison between DAVIS 2016 and DAVIS 2017 datasets. The former (left) has a single mask per group of objects as the latter (right) has a mask for each object to segment.

All the training phases in this thesis use the original DAVIS 2017 training set, while evaluation is made under both the validation and test-dev sets.

In addition to all the annotated videos, DAVIS provides a benchmark [23] based on two metrics. On the one hand, the mean *Jaccard* distance ($\mathcal{J}$ mean) also called mean Intersection over Union ($mIoU$), and on the other hand, the *F-score* ($\mathcal{F}$) over the object boundaries.

For each frame, the $\mathcal{J}$ mean is calculated by averaging the individual *Jaccard* distances for all the present objects. For each object, the IoU is the ratio between the intersection area and the area of union of the ground truth and the predicted masks. Alternatively, for each object,

$$\mathcal{J} = IoU = \frac{TP}{TP + FP + FN} \tag{3.1}$$

where TP (True Positives) is the number of pixels that were predicted as part of the object, and they really belonged to it; FP (False Positives) is the number of pixels that were predicted but did not belong to the object; and FN (False Negatives) is the number of object pixels that were not predicted.

$$IOU = \frac{\text{area of overlap}}{\text{area of union}} =$$

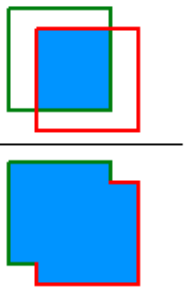


Figure 3.3: Visual representation of Intersection over Union where the green square acts as the ground truth mask, the red one as the prediction, and the areas of interest are filled in blue.

The *F-score*, by its side, is a ratio between Precision (Pr) and Recall (Rc). The former calculates how many of the predicted pixels are relevant while the latter measures how many of the relevant items have been predicted:

$$\mathcal{F} = 2 \cdot \frac{Pr \cdot Rc}{Pr + Rc} \tag{3.2a}$$

$$Pr = \frac{TP}{TP + FP} \tag{3.2b}$$

$$Rc = \frac{TP}{TP + FN} \tag{3.2c}$$

In the DAVIS benchmark, the *F-score* is measured on the masks' boundaries, which means that in this case, TP is the number of pixels that were predicted as belonging to the boundary of the object mask, and they indeed were. The equivalent is applied to FP and FN definitions.

A boundary is defined as the pixel were an object ends, or another one starts. Predicting the exact border pixel in a 480p image is incredibly difficult. For this reason, the DAVIS boundary *F-score* has a certain margin (around 8 pixels), and any predicted edge pixel that is located inside this margin is computed as correct.

Finally, DAVIS benchmark defines an additional metric to summarize the accuracy of a method, the $\mathcal{G}$ mean. It is calculated as the mean between the $\mathcal{J}$ mean and the *F-score*:

$$\mathcal{G} = \frac{\mathcal{J} + \mathcal{F}}{2} \tag{3.3}$$

The global benchmark for a specific method on a video set (training, validation or test-dev) is obtained as the mean of all the sequences in it, while the values for each sequence are obtained averaging the scores of each of the individual frames, which at the same time are calculated averaging the scores for every object in them.

3.3 Previous work on Video Object Segmentation

Since now, there have been many different approaches to Video Object Segmentation. Although all of them are different, there are two main basic ideas: Object Detection and Mask Propagation.

Object Detection methods mostly rely on the first frame, learning the appearance of the object to be segmented from the given masks and trying to detect it in the following frames. They process video sequences as individual images, without using temporal consistency, so they usually suffer from appearance changes in non-solid objects (*e.g.* people or animals), scale and pose variations.

Caelles *et al.* in OSVOS [2] online fine-tune their model for each sequence, in order to learn the exact appearance of the object to be separated from its background. Then, they process the other frames one by one, without using temporal information. In OSVOS-S [22], they incorporate a semantic supervision to their previous method improving their results. As they only use the object appearance of the first frame, they need to train for several iterations until the network learns the specific task of segmenting that particular object. This online fine-tuning step on test time impede both methods to be time efficient.

By its part, Hu *et al.* in VideoMatch [11] also use the first frame exclusively but in a different manner. They extract features from it for both the objects and the background to then compare them to the extracted features of each other frame. Additionally, Voigtlaender *et al.* in OnAVOS [30] implement the ability to adapt and update the reference appearance of the objects in the first frame to cope with aspect variations while processing the whole sequence. Both methods add online fine-tuning to achieve better results, having the same time efficiency problem.

On the other hand, we can find the Mask Propagation based methods. These models often use optical flow to warp predicted masks of previous frames according to the motion of the video. They rely on temporal consistency, and thus, recursive networks are often used.

Hu *et al.* in MaskRNN [10] use FlowNet 2.0 [12] to extract optical flow between adjacent frames and use it to propagate the first frame ground truth masks through all the video, while Bao *et al.* use a similar idea and incorporate an MRF postprocessing step in CINM [1].

Figure 3.4: Optical Flow generated from FlowNet 2.0 between adjacent frames. Red areas show where movement is while static areas are painted in blue.

Their heavy reliance on the previous frame predicted masks make them vulnerable to occlusions, as when an object disappears they do not have any mechanism to retrieve them. Furthermore, they rely on previous predictions instead of ground truth, which has the risk of degenerating if predictions are not good enough.

As both Image Detection and Mask Propagation have their benefits, it is easy to find some models that combine both of them in order to avoid the problems they have individually.

Oh *et al.* in RGMP [32], for example, use a siamese network where one of its branches is dedicated to a reference frame (the first one), and the other one to the last processed frame. Combining two references gives them some robustness against pose variation and occlusions. SiamMask [31] also uses a siamese network to take advantage of the previously predicted masks. Any of them need online fine-tuning which converts them into fast and time efficient methods.

Furthermore, FAVOS [4], DyeNet [19], and PReMVOS [21] use a Region Of Interest localization network (such as Mask-RCNN [8]) to retrieve new objects that appear in farther frames and compare them to previously stored templates of the objects of interest. This re-identification mechanism helps them when objects are occluded, or significant appearance variations happen. In addition to this, DyeNet and PReMVOS also use optical flow to warp the masks of previously processed frames in accordance with the video movement.

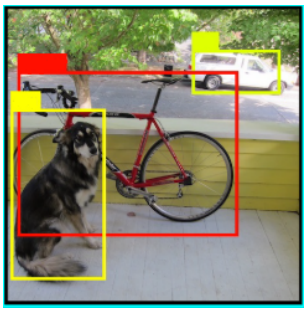


Figure 3.5: Region of Interest localization output with 3 detected objects.

Mixing object detection, re-identification, and mask propagation techniques lead them to state-of-the-art results. However, these are time consumptive methodologies, and combining them with online fine-tuning or postprocessing steps make these models dilatory.

Finally, RVOS [29], recently presented by Ventura *et al.*, is an extremely rapid spatiotemporal RNN architecture where the spatial branch predicts all the objects for a specific time step and the temporal section is which contains all the previous time steps memory. Apart from being a solution to the one-shot problem, it is one of the first approaches to face the zero-shot unsupervised task.

4. Fast Video Object Segmentation by Pixel-Wise Feature Comparison

This chapter is meant to be the principal episode of this thesis because it contains the presented solution to Video Object Segmentation. After a brief explanation of the idea behind it, this chapter also includes an overview of the proposed model and a complete description of each of its individual components. Finally, it also contains some information about the training schedule followed to lead the model to state-of-the-art results.

4.1 The idea

Nowadays, 3 hours of video are uploaded to YouTube every second. We would need more than 10000 humans working all day, every day to process, tag and classify this enormous amount of data, so it is evident that we need algorithms for machines to do this work. Furthermore, video has become more and more important due to growing technologies such as autonomous driving.

Many researching groups have detected this need for developing useful VOS methods. However, most of them end up with very complex networks and algorithms that often need tens or hundreds of seconds to process each photogram.

The goals of Fast VOS by Pixel-Wise Feature Comparison (PiWiVOS) are to be simple and fast enough to be used in Real-Time applications. Furthermore, in the same aim to be used in real-world environments, our method is directly addressed to the multi-object VOS problem. One can easily see that almost all the existing VOS models, presented in Section 3.3, start solving the foreground-background segmentation problem, to then move to the multi-object task by merging their results. Even those methods that are prepared and trained for multi-object segmentation directly, need a forward pass for each of them or an RNN to predict them one by one. All the previously presented approaches are object-wise methods.

In contrast, our model PiWiVOS is a pixel-wise approach, where with a single forward pass, pixel embeddings are extracted and compared with previously obtained features of two different references: the first frame (where ground truth masks are given) and the predictions of the previous time step. This simplistic approach leads our method to state-of-the-art results in both its accuracy and its processing speed.

4.2 The Implementation

4.2.1 Definitions

Let $I = \{I^0, I^1, \ldots, I^t, \ldots, I^{T-1}\}$ be the given sequence of T RGB images. Furthermore, let $\hat{y}^t \in \{0, 1, \ldots, n, \ldots, N-1\}^{H \times W}$ be the prediction for frame t where H and W are the original height and width of the video, and n denotes the predicted object for each pixel among the N possible objects, including background, which is represented as $n = 0$. Finally, let y^0 be the given ground truth masks for the first frame.

4.2.2 Model Overview

PiWiVOS has two parallel branches, each of them processing the current frame and comparing it to a particular reference frame. The first section uses the first frame, I^0 (where ground truth masks are given y^0) as guidance, while the second branch matches the processed frame to the previous one, I^{t-1}, using the earlier predicted masks $\hat{y}^{t-1}$. Finally, the result of both branches are combined to produce the output predictions. Both branches are exactly the same and share the CNN weights.

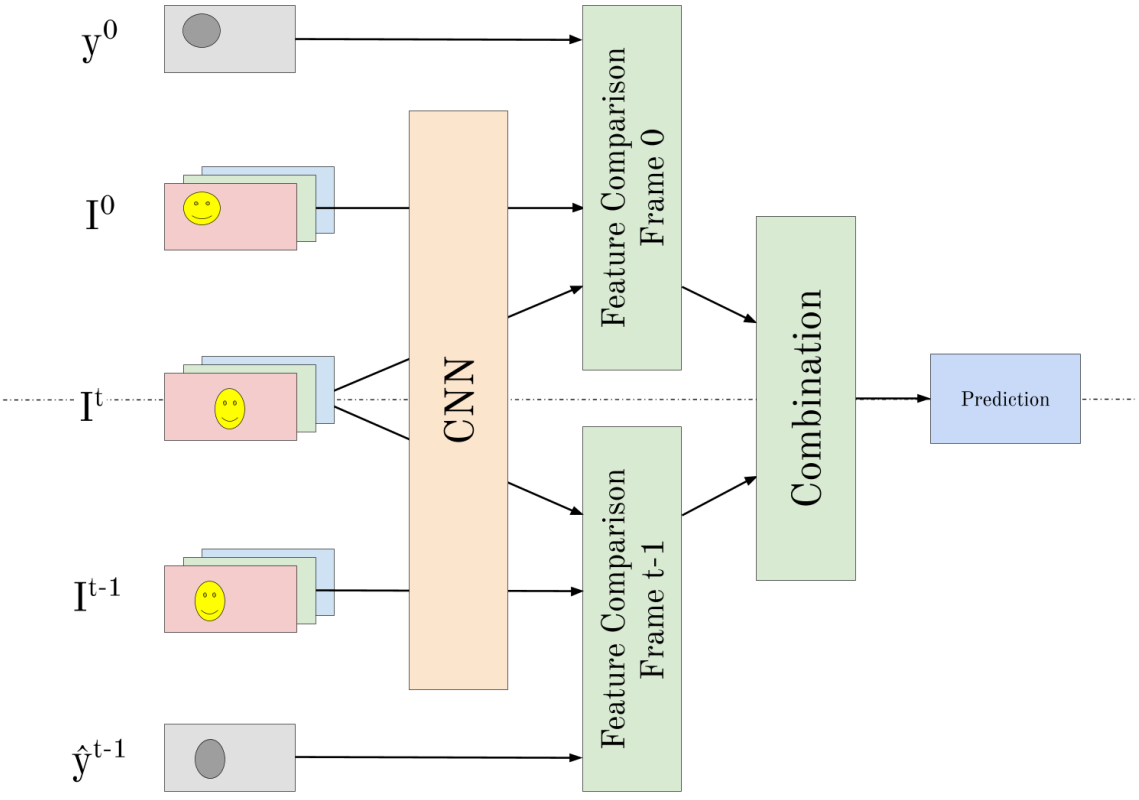


Figure 4.1: PiWiVOS' Overview.

The inputs of each branch are two RGB images, I^t, and $I^{\text{reference}}$, which is either I^0 or I^{t-1}. Each image is individually forward-passed through a feature extractor, generating a low-resolution feature volume. Both feature volumes are compared with the guidance of the reference masks (either y^0 or $\hat{y}^{t-1}$), and a pixel-wise object score map is generated, where every pixel in the low-resolution dimensions has a score list of belonging to one object or the other.

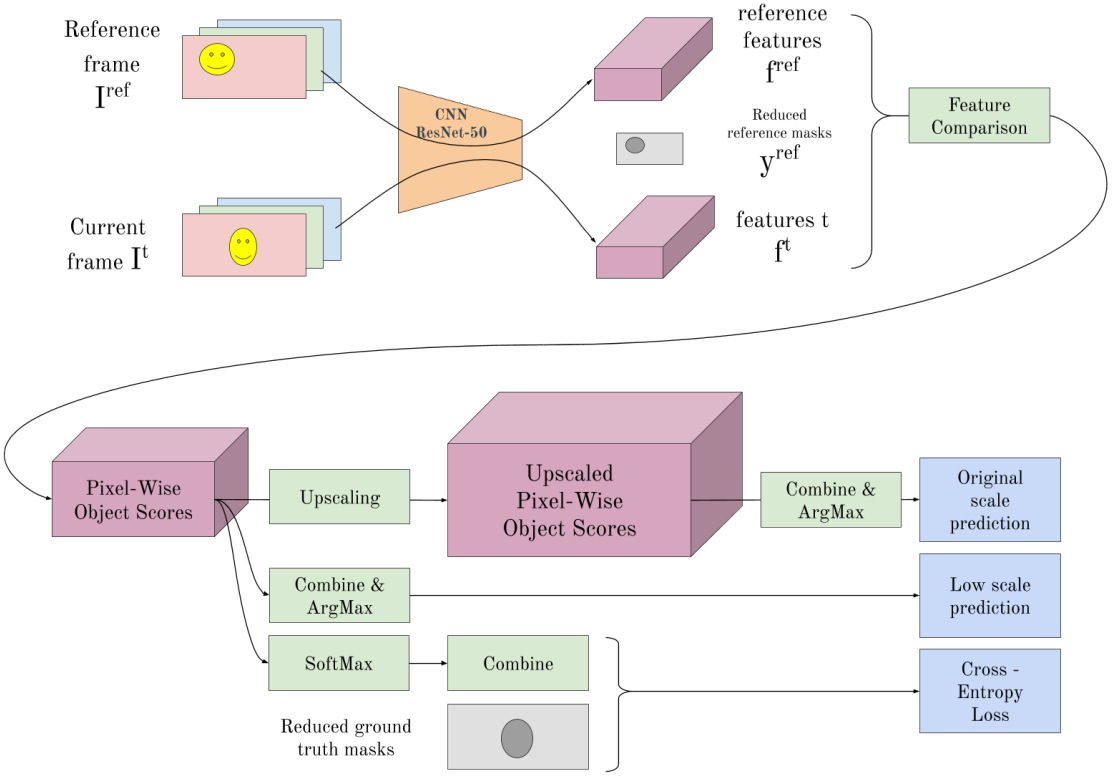


Figure 4.2: A single branch of the PiWiVOS' pipeline.

This score volume is then used to generate three different outputs. Firstly, it is bilinearly upscaled to the dimensions of the input image, and combined with the other branch upscaled volume. The highest scored object (pixel-wise) is chosen to be the final prediction. Secondly, the low-resolution score volume is also combined with the other branch's scores to output a low-scale prediction, which will be used in the subsequent time step as reference masks in the comparison module. Finally, when training, these scores are normalized into probabilities using the SoftMax function, combined with the other branch object probabilities, and lastly compared to the ground truth by the loss function.

Hereafter, each component of the method will be individually presented and described.

4.2.3 Individual Components' Description

Data Preprocessing

DAVIS 2017 videos are given as sequences of individual RGB images with 8-bit codification for each of their 3 channels. They are provided on both Full-Resolution and 480p, although the benchmarks are calculated on the 480p predictions. In order to use these images with a CNN, a linear normalization is performed in each of their three channels, re-scaling their values from $[0, 255]$ to $[0, 1]$.

On the other side, the ground truth masks are given with a single-channel palette-mode image where each integer value represents an object: pixels with the value 0 belong to background, those with value 1 are classified as *object1*, and so on. The ground truth masks do not need any preprocessing.

The predictions must be exported with the same format than the ground truth masks to use the DAVIS' automatic benchmarking tool.

The Feature Extracting Network

The chosen architecture for feature extracting is ResNet-50, which is a 50-layer reduced version of the original ResNet, which has 152. As explained in Section 2.3.1, CNN's tend to reduce the spatial dimensions while increasing the number of channels or feature maps. If we obviate the final fully connected layer, ResNet-50 outputs 2048 channels where the height and width have been reduced by a factor of 32 in respect to the input image. Although this is acceptable when performing image classification, in segmentation problems we require a dense output as each pixel has to be labeled.

PiWiVOS uses the first 4 layer blocks of ResNet-50 (see Figure 4.3), from conv1 to conv4_x. Furthermore, the stride of the convolutional layers in the last block (conv4_x) is reduced from 2 to 1 to have four times the number of outputted pixels. The resulting extracted volume consists of 1024 features with a reduction of 1/8 in both the height and the width in respect with the original input image. It is a perfect balance between the number of features, their receptive field, and the output dimensions for a segmentation task.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 4.3: ResNet versions and their layers. Note that output size is relative to the input.

The Feature Comparison Module

The feature comparison module is the most essential part of the method because it is the one that enables the pixel-wise approach and permits to generate a multi-object prediction by only forward-passing each frame once.

Their inputs are the extracted feature volume for the current frame, $f^t \in \mathbb{R}^{h\times w\times 1024}$ (where h and w are the reduced height and width respectively), the feature volume for the reference frame $f^{\text{ref}} \in \mathbb{R}^{h\times w\times 1024}$, and the reduced reference masks $y^{\text{ref}} \in \{0, 1, \ldots, N-1\}^{h\times w}$. Note that the reference can either be the first frame or the previous one, where y^{ref} would become $\hat{y}^{t-1}$. For the rest of the explanation we will use y^{ref} as both the ground truth for frame I^0 and the predicted masks for frame I^{t-1}.

Once we have the inputs, for each pixel i in the reduced dimensions of the current frame t, we take its feature vector $f_i^t \in \mathbb{R}^{1024}$ and we compare it with the feature vector f_j^{ref} of every pixel j in the reference, generating a similarity score $s_{i,j}^{\text{ref}}$:

$$s_{i,j}^{\text{ref}} = \text{similarity}(f_i^t, f_j^{\text{ref}}) \qquad \text{for all } j \in \{0, 1, \ldots, h\cdot w - 1\} \tag{4.1a}$$

$$\text{similarity}(f_i^t, f_j^{\text{ref}}) = \frac{{f_i^t}^\top \cdot f_j^{\text{ref}}}{\|f_i^t\| \cdot \|f_j^{\text{ref}}\|} - \lambda \cdot \text{distance}(i,j) \qquad \lambda \in [0,1] \tag{4.1b}$$

$$\text{distance}(i,j) = \frac{\sqrt{(i_x - j_x)^2 + (i_y - j_y)^2}}{\sqrt{(h-1)^2 + (w-1)^2}} \tag{4.1c}$$

where λ is a parameter that penalizes the possible fact that two separate pixels have similar features, and x and y are the Cartesian coordinates for the pixel of interest inside the two-dimension reduced image.

After comparing pixel i with each of the pixels j in the reference frame, these similarity scores $s_{i,j}^{\text{ref}}$ are converted to a single vector of object scores $\boldsymbol{o_i^{\text{ref}}}$:

$$\boldsymbol{o_i^{\text{ref}}} = (o_{i,0}^{\text{ref}}, o_{i,1}^{\text{ref}}, \ldots, o_{i,n}^{\text{ref}}, \ldots, o_{i,N-1}^{\text{ref}}) \tag{4.2}$$

where each of these entries contains the score of pixel i to belong to object n and is calculated as follows:

$$o_{i,n}^{\text{ref}} = \text{avg}\left(\underset{j|y_j^{\text{ref}}=n}{\text{topK}}\ (s_{i,j}^{\text{ref}})\right) \qquad \text{for all } n \in \{0,1,\ldots N-1\} \tag{4.3}$$

where the topK operator selects the K highest scored $s_{i,j}^{\text{ref}}$. The value of the parameter K, which allows avoiding outliers, will be discussed in Section 5.1.

All these calculations are made for each pixel $i \in \{0,1,\ldots,h \cdot w - 1\}$. When put together, it results in an object scores volume of dimensions $h \times w \times N$.

Object Scores merging

After calculating the object scores with the frame I^0 as a reference, $\boldsymbol{o_i^0}$, and those with the previous frame, $\boldsymbol{o_i^{t-1}}$, both of them are combined to produce three outputs.

To calculate the final prediction, each of these object score volumes is upscaled to the dimensions of the input images (note that the dimensions had been reduced inside the CNN) by bilinear interpolation. Afterward, for each pixel i, both object score vectors, $\boldsymbol{o_i^0}$ and $\boldsymbol{o_i^{t-1}}$, are combined by taking the maximum score object-wise, resulting in $\boldsymbol{o_i}$. The predicted object is chosen by taking the index of the highest score in the array

$$\boldsymbol{o_i} = (o_{i,0}, o_{i,1}, \ldots, o_{i,n}, \ldots, o_{i,N-1}) \tag{4.4a}$$

$$o_{i,n} = \max(o_{i,n}^0, o_{i,n}^{t-1}) \qquad \text{for all } n \in \{0,1,\ldots,N-1\} \tag{4.4b}$$

$$\hat{y}_i = \underset{n}{\text{argmax}}(o_{i,0}, o_{i,1}, \ldots, o_{i,n}, \ldots, o_{i,N-1}) \tag{4.4c}$$

The same process is followed to generate the low-scale prediction (without upscaling) that will be used in the succeeding time step.

To produce the last output, however, it is crucial to take into consideration that it is not the same combining the scores to produce a prediction, than using them to calculate the loss. In this last case, the loss needs to be fed normalized probabilities in order to work correctly. For this reason, each object score vector is normalized by the SoftMax function ($\sigma()$) separately. Then, for each pixel i both probability vectors are combined in the following weighted sum:

$$p_{i,n} = w_{\text{comb}} \cdot \sigma(\boldsymbol{o_i^0})_n + (1 - w_{\text{comb}}) \cdot \sigma(\boldsymbol{o_i^{t-1}})_n \qquad \text{for all } n \in \{0, 1, \ldots, N-1\} \tag{4.5}$$

Weighted multi-class Cross-Entropy Loss

As seen in Section 2.2.2, Binary Cross-Entropy Loss is the most utilized cost function in classification problems where the target has two possible discrete values, either cats and dogs or 0 and 1. Its operation is simple: in this last example, for the case when the ground truth is 1, the predicted probability is "pushed" towards 1 proportionally to the logarithmic distance between the prediction and the target value, 1 (see Equation 2.4).

The multi-class scene does not differ that much from the binary problem. The main difference is that it does not have a single predicted value anymore, but a prediction for each of the objects or categories. As a consequence, the multi-class Cross-Entropy Loss for N objects, where the predicted probabilities are p_n for $n \in \{0, 1, \ldots, N-1\}$, and the correct object is n^*, looks like:

$$\mathcal{L}_{CE} = -\log(p_{n^*}) \tag{4.6}$$

Note that although it seems that the loss function is only working on p_{n^*}, in addition to pushing it to 1, it is implicitly pushing all the other probabilities $p_{n|n\neq n^*}$ to 0 as they must add to 1.

In our pixel-wise approach, the Cross-Entropy Loss is applied to all the pixels individually, to be then averaged for the whole frame, which has a potential problem.

Let's imagine a scenario where all frames contain one single object apart from the background. If these frames have an average of 90% of background pixels, and only 10% of its pixels belong to the object, the network can eventually learn to predict everything as background, and stabilize with a quite small loss. This situation is prevalent in the DAVIS dataset sequences, where this behavior is not desired as all the objects are equally important.

A weighted version of the multi-class Cross-Entropy Loss is proposed to solve this problem:

$$\mathcal{L}_{CE} = -w_{n^*} \log(p_{n^*}) \tag{4.7a}$$

$$w_n = \frac{\dfrac{1}{|\{y|y=n\}|}}{\displaystyle\sum_{k=0}^{N-1} \frac{1}{|\{y|y=k\}|}} \qquad \text{for all} \quad n \in \{0, 1, \ldots, N-1\} \tag{4.7b}$$

where $|\{y|y=n\}|$ is the number of ground truth pixels labeled as object n in the whole frame.

Thereby, in our model, the following equation is used to calculate the loss for each frame:

$$\mathcal{L} = \underset{i}{\text{avg}}(\mathcal{L}_i) = \underset{i}{\text{avg}}(-w_{n_i^*} \log(p_{i,n_i^*})) \tag{4.8}$$

where n_i^* is the correct object for each pixel i.

4.2.4 Training Schedule

This model is trained using ResNet-50 pre-trained weights on ImageNet directly downloaded from the torchvision model zoo [25]. From this starting point, it is then fine-tuned on the DAVIS 2017 training dataset using the Stochastic Gradient Descent Momentum optimizer, with hyperparameters $\lambda = 5e^{-4}$ and $\beta = 0.9$.

The model is trained for 30 epochs during 12 hours on a single Nvidia Titan XP GPU, using a mini-batch size of 1 due to memory limitations (each batch contains the processed frame, its previous frame and the first frame of the sequence).

The implementation of PiWiVOS, coded in Python using the PyTorch library, has been published on https://github.com/rafelps/PiWiVOS.

5. Experiments and Results

This chapter contains all the experiments and results that empirically substantiate each of the parameters' value and component design. Furthermore, it also includes an ablation study on the performance of each component of the method, and a comparison of the accuracy and speed of the most relevant Video Object Segmentation approaches.

5.1 Experiments on the model's components

In this section, we discuss the decision of selecting each of the chosen components and parameters. In each of the individual experiments, all the parameters that are not analyzed are kept constant.

The Feature Extracting Network

It is well known that deeper networks usually have better results due to their larger capacity to adapt to new data. However, more layer implies more parameters and more computation resources, both time and memory. In addition to the number of layers, the stride factor in the conv4_x layer changes the outputted feature volumes. When the stride is 2, the dimensions of the input image are reduced by a factor of 16 in both the height and the width. However, when the stride is set to 1, the reducing factor becomes 8, so the output becomes 4 times larger, which makes the method slower as it has to process larger tensors. In the following experiment, we compare different versions of ResNet to choose the one that best fits our objectives.

ResNet version	conv4_x stride	$\mathcal{J}$ Mean	$\mathcal{F}$ Mean	$\mathcal{G}$ Mean	FPS
ResNet-34	2	55.7	54.0	54.9	**85**
ResNet-50	2	56.6	54.7	55.7	56
ResNet-101	2	59.4	56.2	57.8	35
ResNet-34	1	64.3	70.7	67.5	26
ResNet-50	1	67.6	74.6	**71.1**	18

Table 5.1: $\mathcal{J}$ mean, $\mathcal{F}$ mean, $\mathcal{G}$ mean and FPS for different ResNet versions and configurations on DAVIS 2017 validation set.

There is a clear relationship between the output dimensions (higher outputs lead to more resolution in our predictions), the number of layers, the model accuracy, and its speed: more computation means better results but a slower method. However, the slowest solution, ResNet-50 (conv4_x stride 1), is already faster than almost every other published method, achieving state-of-the-art results. For this reason, it is the chosen backbone for PiWiVOS.

Nevertheless, our approach is fully compatible with all the presented versions of ResNet. Therefore, for other applications that do not need state-of-the-art accuracy, ResNet-34 (conv4_x stride 2) could also be an interesting choice as it is 1500 times faster than other methods with similar accuracy. We will call this approach PiWiVOS-Fast.

The Feature Comparison Module

The first parameter to be analyzed in the feature comparison module is λ, which penalizes the possible fact that two separate pixels have similar features (see Equation 4.1b). It may be interesting to use different λ's for each branch, so the following values for λ^0 and λ^{t-1} have been studied:

λ^0	0	0.25	0.5	0.75	1
$\mathcal{J}$ Mean	64.0	67.4	**67.6**	67.0	66.1

Table 5.2: $\mathcal{J}$ mean for different values of λ^0 on DAVIS 2017 validation set.

λ^{t-1}	0	0.25	0.5	0.75	1
$\mathcal{J}$ Mean	63.4	66.8	**67.6**	**67.6**	67.3

Table 5.3: $\mathcal{J}$ mean for different values of λ^{t-1} on DAVIS 2017 validation set.

It can easily be seen that in both branches the λ penalization improves the accuracy as the worst results happen when $\lambda = 0$. Furthermore, the model seems to work better with $\lambda^0 \leq \lambda^{t-1}$, which also makes sense. Video sequences contain continuous motion, and therefore objects in adjacent frames (t, t-1) have to be relatively close, or at least closer than when comparing the last frames with the first one, where it is possible that objects have different positions.

Following, there is an analysis of the parameter K, presented in Equation 4.3. Analogously as with λ, different values of K may work on each branch. For this reason, parameters K^0 and K^{t-1} are defined.

K^0	1	2	3	5	10	all
$\mathcal{J}$ Mean	**67.6**	67.1	66.8	65.7	64.1	52.3

Table 5.4: $\mathcal{J}$ Mean for different values of K^0 on DAVIS 2017 validation set.

K^{t-1}	1	3	5	10	15	all
$\mathcal{J}$ Mean	52.4	64.0	66.0	**67.6**	66.9	54.3

Table 5.5: $\mathcal{J}$ Mean for different values of K^{t-1} on DAVIS 2017 validation set.

The results of this experiment are also logical. In the first branch, where the reference is the ground truth of the first frame, the calculated similarity scores $s^0_{i,j}$ are very reliable, so the best performance is achieved when K is set to 1. In this case, the object score $o^0_{i,n}$ takes the information of one single pixel in the reference frame:

$$o^0_{i,n} = \text{avg}\left(\underset{j|y^0_j=n}{\text{top1}}\,(s^0_{i,j})\right) = \text{avg}\left(\max_{j|y^0_j=n}(s^0_{i,j})\right) = \max_{j|y^0_j=n}(s^0_{i,j}) \tag{5.1}$$

However, in the second branch, ground truth is not available and we work on previous predictions. In this case, a value of 10 works the best because when averaging scores between

pixel i and 10 different pixels j that belong to object n on the reference frame I^{t-1}, the variance of the prediction is reduced.

$$o_{i,n}^{t-1} = \text{avg} \left(\underset{j|\hat{y}_j^{t-1}=n}{\text{top10}} \; (s_{i,j}^{t-1}) \right) \tag{5.2}$$

Note that when K is set to *all*, the performance drops. In this case, the operator topK does not intervene, so the object score $o_{i,n}^{\text{ref}}$ is calculated by averaging all the similarity scores between pixel i and every pixel j in the reference frame that belongs to object n. This does not work because not all the pixels of an object have the same features.

$$o_{i,n}^{\text{ref}} = \text{avg} \left(\underset{j|y_j^{\text{ref}}=n}{\text{top}all} \, (s_{i,j}^{\text{ref}}) \right) = \underset{j|y_j^{\text{ref}}=n}{\text{avg}} \; (s_{i,j}^{\text{ref}}) \tag{5.3}$$

In addition to all the results presented in Tables 5.2, 5.3, 5.4 and 5.5, many experiments were made with different combinations of values for K^0, K^{t-1}, λ^0 and λ^{t-1}. They seem to work quite independently, and the best performance is achieved with the following combination:

$$K^0 = 1 \qquad K^{t-1} = 10 \qquad \lambda^0 = 0.5 \qquad \lambda^{t-1} = 0.5$$

Object Scores merging

Several options have been studied to merge the two branches' object score volumes:

- **Option A:** $o_{i,n} = \max(o_{i,n}^0, o_{i,n}^{t-1})$
- **Option B:** $o_{i,n} = w \cdot o_{i,n}^0 + (1-w) \cdot o_{i,n}^{t-1}$
- **Option C:** $o_{i,n} = \sigma(o_{i,0}, \dots o_{i,N-1})_n$ where $o_{i,n} = \max(o_{i,n}^0, o_{i,n}^{t-1})$
- **Option D:** $o_{i,n} = w \cdot \sigma(o_{i,0}^0, \dots, o_{i,N-1}^0)_n + (1-w) \cdot \sigma(o_{i,0}^{t-1}, \dots, o_{i,N-1}^{t-1})_n$

where $\sigma()$ is the SoftMax function described in Equation 2.2, and $w \in [0, 1]$ is a weight.

Option	A	B $w = 0.25$	B $w = 0.5$	B $w = 0.75$	C	D $w = 0.25$	D $w = 0.5$	D $w = 0.75$
$\mathcal{J}$ Mean	**67.6**	61.2	62.0	62.7	63.4	61.4	61.5	62.7
FPS	**18**	18	18	18	14	14	14	14

Table 5.6: $\mathcal{J}$ mean and FPS for different branch merging options for output predictions on DAVIS 2017 validation set.

It results that the best option is to use the max function instead of a weighted sum, and the raw scores better than normalized ones, which is an advantage for the time efficiency because calculating the SoftMax function pixel-wise carries a slowdown of 4 FPS.

However, when we have to merge the results to feed the loss function, they have to be normalized, so a different analysis is made for this particular case with options C and D.

Option	C	D $w_{comb} = 0$	D $w_{comb} = 0.25$	D $w_{comb} = 0.5$	D $w_{comb} = 0.75$	D $w_{comb} = 1$
$\mathcal{J}$ Mean	66.7	66.5	66.7	66.9	67.5	**67.6**

Table 5.7: $\mathcal{J}$ mean for different branch merging options for loss function on DAVIS 2017 validation set.

In this case, the best option to feed the loss function is to directly give it the normalized vector of object scores from the first branch, where the reference is the first frame and we have the ground truth masks available. We can interpret that predictions coming from the first branch are more reliable and stable, so they lead to a better training. Note that this does not mean that the second branch is dispensable because their scores are still being used to calculate the final predictions, as will be seen in Section 5.2.

Weighted multi-class Cross-Entropy Loss

Finally, a simple study on the proposed Weighted multi-class Cross-Entropy Loss has been made:

Cross-Entroy Loss	Weighted	Non-Weighted
$\mathcal{J}$ Mean	**67,6**	66,9

Table 5.8: $\mathcal{J}$ mean for the different options for loss function on DAVIS 2017 validation set.

It can be seen that the weighted loss gives the model a boost in performance, which is logical as it helps to give each object the same importance.

5.2 Ablation Study on the model's components

In this section, an ablation study has been performed to confirm the effectiveness of each component of our method. Pre-trained experiments use the pre-trained weights of ResNet on Imagenet, without fine-tuning on DAVIS, while $-K$ and $-\lambda$ mean using the values 1 and 0 respectively instead of the optimal ones, and $-$branch means that this part of the model is not used.

Configuration	$\mathcal{J}$ Mean
Pre-trained $-$branch 0	47.4
Pre-trained $-$branch $t-1$	52.4
Pre-trained $-K$ $-\lambda$	47.4
Pre-trained $-K$	47.6
Pre-trained $-\lambda$	52.8
Pre-trained	59.3

Configuration	$\mathcal{J}$ Mean
PiWiVOS $-$branch 0	58.8
PiWiVOS $-$branch $t-1$	57.3
PiWiVOS $-K$ $-\lambda$	54.3
PiWiVOS $-K$	54.5
PiWiVOS $-\lambda$	63.8
PiWiVOS	**67.7**

Table 5.9: Ablation Study on the model's components.

It can be seen that each of the method's components, including the training schedule, is crucial to reach state-of-the-art performance. The topK operator is the one that gives the most significant boost (+13.2%), while the distance penalization λ gives the least (+3.9%). In addition to this, the fine-tuning stage on DAVIS training set improves the method's accuracy approximately 10%. Finally, note that although outputs of the $t-1$ branch are not used in the

loss function, the performance drops a significant 10.4% when not using them to make the final predictions.

5.3 Comparison of the state-of-the-art VOS methods

In this section, our presented solution to Video Object Segmentation, PiWiVOS, is compared to the state-of-the-art approaches to the task, taking in consideration both the accuracy and the speed of them.

Method	$\mathcal{J}$ Mean	$\mathcal{F}$ Mean	$\mathcal{G}$ Mean	FPS
PReMVOS [21]	**73.9**	**81.7**	**77.8**	0.025
DyeNet [19]	-	-	74.1	0.2
CINM [1]	67.2	74	70.6	0.01
OSVOS-S [22]	64.7	71.3	68	0.11
OnAVOS [30]	61.6	69.2	65.4	0.035
OSVOS [2]	56.6	64.0	60.3	0.05
FAVOS [4]	54.6	61.8	58.2	0.3
RGMP [32]	64.8	68.6	66.7	3
PReMVOS$^-$ [21]	-	-	65.7	1.25
RVOS [29]	57.5	63.7	60.6	22
VM [11]	56.5	-	-	1.43
OSMN [34]	52.5	57.1	54.8	4
SiamMask [31]	51.1	55	53.1	17.5
PiWiVOS-F	55.7	54	54.9	**85**
PiWiVOS	**67.6**	**74.6**	**71.1**	18

Table 5.10: $\mathcal{J}$ mean, $\mathcal{F}$ mean, $\mathcal{G}$ mean and FPS comparison between the state-of-the-art VOS methods on DAVIS 2017 validation set. The methods above use online fine-tuning while the approaches on the bottom do not. ($^-$) denotes the offline version of a fine-tuned method. The speed of those object-wise methods that only present their FPS on DAVIS 2016 is divided by a factor of 2 because DAVIS 2017 validation set has an average of two objects per frame.

As seen in Table 5.10, PiWiVOS outperforms all non-fine-tuned methods while working faster than them. Furthermore, PiWiVOS' results can also be compared to the online fine-tuned methods. Although these approaches work between 60 and 1800 times slower, only two of them can achieve better performance.

The same comparison is shown in Table 5.11, for the test-dev set of DAVIS 2017. In this case, similar results are obtained. PiWiVOS achieves state-of-the-art results among the non-fine-tuned methods while running faster than them (note that DyeNet$^-$ does not use online fine-tuning, but its strong postprocessing makes it as slow as fine-tuned methods). Furthermore, PiWiVOS achieve comparable performance to some of the fine-tuned methods.

Finally, Figure 5.1 shows a chart where the different methods are compared. There is a clear division between fine-tuned methods and those that do not rely on online learning. Furthermore, in the semi-log axis figure, there is a linear tendency, where slower methods have better $\mathcal{J}$ Mean and vice versa. PiWiVOS outstands at the top-right of this tendency, performing better than similar speed methods, and faster than comparable accuracy ones.

Method	$\mathcal{J}$ Mean	$\mathcal{F}$ Mean	$\mathcal{G}$ Mean	FPS
PReMVOS [21]	**67.5**	**75.7**	**71.6**	0.016
DyeNet [19]	65.8	70.6	68.2	0.13
CINM [1]	64.5	70.5	67.5	0.01
OSVOS-S [22]	52.9	62.1	57.5	0.07
OnAVOS [30]	49.9	55.7	52.8	0.023
OSVOS [2]	47.0	54.8	50.9	0.033
FAVOS [4]	42.9	44.3	43.6	0.2
DyeNet$^-$ [19]	**60.2**	**64.8**	**62.5**	0.8
RGMP [32]	51.3	54.3	52.8	2
RVOS [29]	48	57.2	52.6	22
SiamMask [31]	40.6	45.8	43.2	11.66
OSMN [34]	37.7	44.9	41.3	2.33
PiWiVOS-F	42.5	44.4	43.5	**85**
PiWiVOS	**49.9**	**58.7**	**54.3**	18

Table 5.11: $\mathcal{J}$ mean, $\mathcal{F}$ mean, $\mathcal{G}$ mean and FPS comparison between the state-of-the-art VOS methods on DAVIS 2017 test-dev set. The methods above use online fine-tuning while the approaches on the bottom do not. ($^-$) denotes the offline version of a fine-tuned method. The speed of those object-wise methods that only present their FPS on DAVIS 2016 is divided by a factor of 3 because DAVIS 2017 test-dev set has an average of two objects per frame.

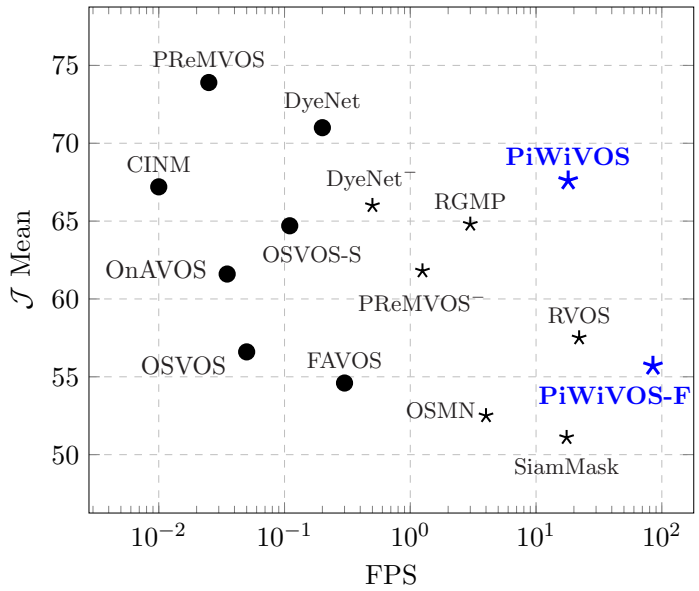


Figure 5.1: Study of methods' accuracy and speed on DAVIS 2017 validation set. The mark ● is used for fine-tuned methods, while ⋆ refers to non-fine-tuned approaches. ($^-$) denotes the offline version of a fine-tuned method. The speed of those object-wise methods that only present their FPS on DAVIS 2016 is divided by a factor of 2 because DAVIS 2017 validation set has an average of two objects per frame.

Further detailed results on the DAVIS 2017 dataset can be seen on appendices B, C and D.

6. Conclusions

This chapter, the last one of this thesis, contains a summary of the whole project work along with some ideas for future work.

To begin with, the objective of learning how Artificial Intelligence works has been accomplished, and a brief proof of it is shown in Chapter 2. In addition to it, another fundamental part of the project has been to understand the task of Video Object Segmentation, alongside with the current datasets and benchmarks. Reading all the literature of the specific task was also crucial to recognize that there was an emptiness between the state-of-the-art approaches: almost none of them was tackling the multi-object problem straightforward with high processing speed as a requirement.

All this information, presented in Chapter 3, lead us to the idea of the project, which was converted to the main goals of it: our model had to be simple, and face the multi-object problem taking into consideration both the accuracy and its time efficiency, aiming to be used in Real-Time applications.

Our method, Fast Video Object Segmentation by Pixel-Wise Feature Comparison (PiWiVOS), introduced in Chapter 4, achieves all the proposed objectives. Firstly, it is a pixel-wise approach, and only needs a single forward pass to predict all the object masks of a frame: it solves the multi-object problem straightforward. Secondly, it is a simple method, as only uses a feature extractor, and bases its performance on comparing these features with different reference frames.

Finally, after analyzing the results presented in Chapter 5, we can state that PiWiVOS achieves state-of-the-art results among non-fine-tuned methods while running faster than them. Moreover, it can also be compared with those approaches that use online learning, as only two of them manifest better performance in the DAVIS benchmark, working 90 and 720 times slower respectively.

Regarding the speed of the method, we can think of a possible application in the autonomous driving scene. A car driving at 90km/h advances 25 meters each second. Taking into account that PiWiVOS is able to process 18 frames per second, it means that it can treat an image almost every meter that the car advances. It definitely is a Real-Time approach to Video Object Segmentation and reaches all the proposed goals for the project.

To end up with all the purposes of this thesis, a complete implementation of the model has been published at https://github.com/rafelps/PiWiVOS. Its code is written in the PyTorch library of Python, one of the most used Deep Learning frameworks in the AI world.

Future Work

Although all the objectives of the project have been achieved, there is always room for improvements or new ideas.

DAVIS 2017 is not a huge dataset, and thus, several problems appear when trying to train a model on it. Possibly, the most significant difficulty is that it only has 60 training videos, which are very different between them. This fact provokes that the network has problems to generalize and overfits the training set very quickly. For this reason, it would be very interesting to train on the recently published YouTube-VOS, which is 50 times larger than DAVIS. After being learned the VOS task, it could be fine-tuned and tested on DAVIS using transfer learning.

Furthermore, a possible improvement could be using an independent feature extractor for each of the two branches, where each network could adapt to either the first frame of the previous one. However, this idea could carry some problems regarding computational resources or time efficiency issues.

Taking into account the accuracy achieved by PiWiVOS, it could be a good idea to include some postprocessing steps such as CRF, or superpixel contour merging to study if its performance can be boosted. Nevertheless, although using the same structure, it would be a completely different project because it would lose its main characteristic: its speed.

Bibliography

[1] Linchao Bao, Baoyuan Wu, and Wei Liu. CNN in MRF: video object segmentation via inference in A cnn-based higher-order spatio-temporal MRF. *CoRR*, abs/1803.09453, 2018.

[2] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. One-shot video object segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[3] Sergi Caelles, Alberto Montes, Kevis-Kokitsi Maninis, Yuhua Chen, Luc Van Gool, Federico Perazzi, and Jordi Pont-Tuset. The 2018 davis challenge on video object segmentation. *arXiv:1803.00557*, 2018.

[4] Jingchun Cheng, Yi-Hsuan Tsai, Wei-Chih Hung, Shengjin Wang, and Ming-Hsuan Yang. Fast and accurate online video object segmentation via tracking parts. *CoRR*, abs/1806.02323, 2018.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[10] Yuan-Ting Hu, Jia-Bin Huang, and Alexander Schwing. Maskrnn: Instance level video object segmentation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 325–334. Curran Associates, Inc., 2017.

[11] Yuan-Ting Hu, Jia-Bin Huang, and Alexander G. Schwing. Videomatch: Matching based video object segmentation. *CoRR*, abs/1809.01123, 2018.

[12] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[13] Vicky Kalogeiton, Vittorio Ferrari, and Cordelia Schmid. Analysing domain shift factors between videos and images for object detection. *IEEE transactions on pattern analysis and machine intelligence*, 38(11):2327–2334, 2016.

[14] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). http://www.cs.toronto.edu/ kriz/cifar.html.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[16] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[17] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. , 2010.

[18] Fuxin Li, Taeyoung Kim, Ahmad Humayun, David Tsai, and James M. Rehg. Video segmentation by tracking many figure-ground segments. In *ICCV*, 2013.

[19] Xiaoxiao Li and Chen Change Loy. Video object segmentation with joint re-identification and attention-aware mask propagation. In *ECCV*, 2018.

[20] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[21] Jonathon Luiten, Paul Voigtlaender, and Bastian Leibe. Premvos: Proposal-generation, refinement and merging for video object segmentation. In *Asian Conference on Computer Vision*, 2018.

[22] K.-K. Maninis, S. Caelles, Y. Chen, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. Video object segmentation without temporal information. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2018.

[23] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Computer Vision and Pattern Recognition*, 2016.

[24] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017.

[25] PyTorch. *TorchVision Model Zoo*. https://pytorch.org/docs/stable/torchvision/models.html.

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[28] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[29] Carles Ventura, Miriam Bellver, Andreu Girbau, Amaia Salvador, Ferran Marqués, and Xavier Giró i Nieto. RVOS: end-to-end recurrent network for video object segmentation. *CoRR*, abs/1903.05612, 2019.

[30] Paul Voigtlaender and Bastian Leibe. Online adaptation of convolutional neural networks for video object segmentation. *CoRR*, abs/1706.09364, 2017.

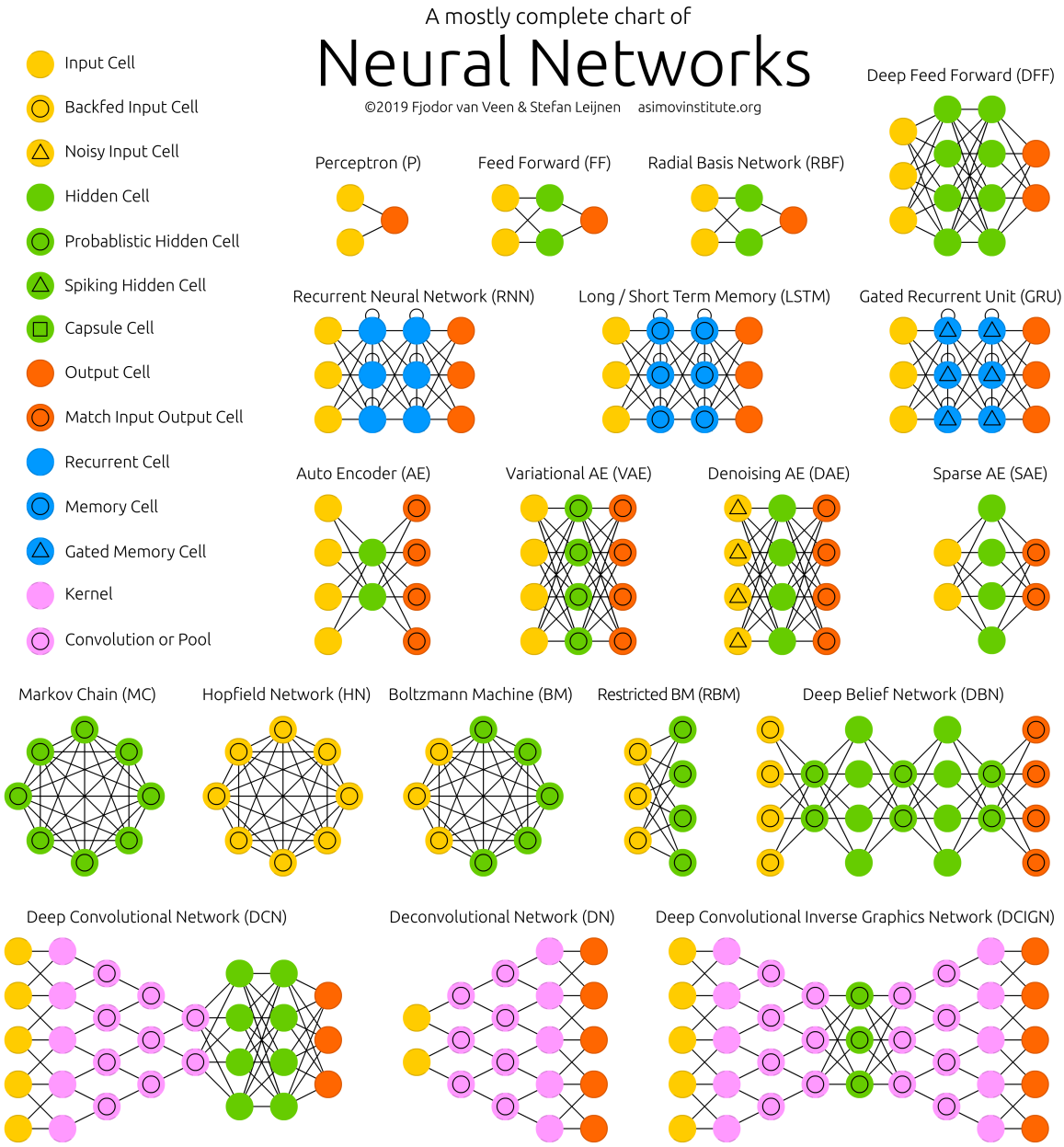
[31] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip H. S. Torr. Fast online object tracking and segmentation: A unifying approach. *CoRR*, abs/1812.05050, 2018.

[32] Seoung Wug Oh, Joon-Young Lee, Kalyan Sunkavalli, and Seon Joo Kim. Fast video object segmentation by reference-guided mask propagation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[33] Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas Huang. Youtube-vos: A large-scale video object segmentation benchmark. *arXiv preprint arXiv:1809.03327*, 2018.

[34] Linjie Yang, Yanran Wang, Xuehan Xiong, Jianchao Yang, and Aggelos K. Katsaggelos. Efficient video object segmentation via network modulation. *CoRR*, abs/1802.01218, 2018.

Appendix A. The Neural Network Zoo



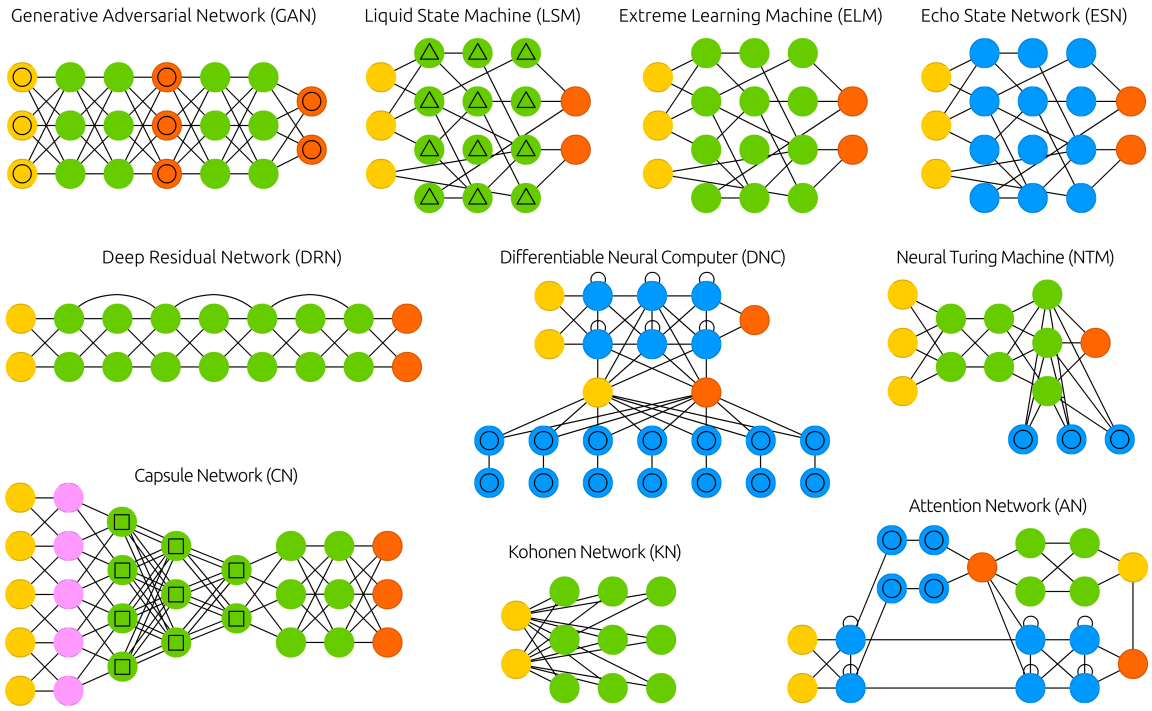


Figure A.1: The Neural Network Zoo.

Appendix B. PiWiVOS' per-object accuracy on DAVIS 2017 validation set

Sequence_Object	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{G}$
bike-packing_1	52.8	64.7	58.8
bike-packing_2	69.4	72.6	71.0
blackswan_1	86.7	93.6	90.2
bmx-trees_1	27.0	66.7	46.9
bmx-trees_2	58.1	77.3	67.7
breakdance_1	75.6	79.4	77.5
camel_1	75.1	86.4	80.8
car-roundabout_1	91.2	86.4	88.8
car-shadow_1	91.6	98.6	95.1
cows_1	90.2	95.5	92.9
dance-twirl_1	75.2	80.4	77.8
dog_1	89.5	90.2	89.9
dogs-jump_1	28.0	40.5	34.3
dogs-jump_2	58.1	65.6	61.9
dogs-jump_3	76.7	91.9	84.3
drift-chicane_1	79.9	84.1	82.0
drift-straight_1	69.8	65.3	67.6
goat_1	80.0	74.6	77.3
gold-fish_1	71.6	64.0	67.8
gold-fish_2	67.0	73.3	70.2
gold-fish_3	72.3	74.6	73.5
gold-fish_4	76.7	79.2	78.0
gold-fish_5	75.8	58.1	67.0
horsejump-high_1	72.7	84.6	78.7
horsejump-high_2	62.9	83.4	73.2
india_1	78.4	73.1	75.8
india_2	56.2	54.5	55.4
india_3	70.9	72.4	71.7
judo_1	71.7	78.6	75.2
judo_2	71.8	73.4	72.6
kite-surf_1	22.0	36.5	29.3
kite-surf_2	28.0	41.6	34.8
kite-surf_3	62.4	87.9	75.2
lab-coat_1	19.6	61.9	40.8
lab-coat_2	0.0	0.0	0.0
lab-coat_3	86.2	81.5	83.9
lab-coat_4	85.0	73.1	79.1
lab-coat_5	82.2	80.0	81.1
libby_1	71.0	90.7	80.9
loading_1	80.9	71.8	76.4
loading_2	52.1	64.9	58.5
loading_3	74.1	82.2	78.2
mbike-trick_1	59.2	77.3	68.3
mbike-trick_2	57.6	61.3	59.5
motocross-jump_1	45.2	56.3	50.8
motocross-jump_2	67.1	64.1	65.6
paragliding-launch_1	73.5	80.2	76.9
paragliding-launch_2	31.2	63.5	47.4
paragliding-launch_3	6.3	26.5	16.4
parkour_1	83.8	91.5	87.7
pigs_1	66.3	64.2	65.3
pigs_2	51.9	66.3	59.1
pigs_3	89.7	83.7	86.7
scooter-black_1	16.6	38.2	27.4
scooter-black_2	68.5	62.5	65.5
shooting_1	30.1	34.8	32.5
shooting_2	78.9	74.0	76.5
shooting_3	68.2	85.6	76.9
soapbox_1	66.9	64.7	65.8
soapbox_2	46.5	54.9	50.7
soapbox_3	24.0	50.2	37.1

Table B.1: PiWiVOS' per-object accuracy on DAVIS 2017 validation set.

Appendix C. PiWiVOS' per-object accuracy on DAVIS 2017 test-dev set

Sequence_Object	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{G}$
aerobatics_1	72.6	66.5	69.6
aerobatics_2	76.0	97.9	87.0
aerobatics_3	73.9	78.1	76.0
car-race_1	59.4	70.0	64.7
car-race_2	90.4	68.7	79.6
car-race_3	37.1	45.7	41.4
car-race_4	51.4	61.3	56.4
carousel_1	35.3	38.4	36.9
carousel_2	22.7	34.5	28.6
carousel_3	26.6	34.3	30.5
carousel_4	40.0	44.5	42.3
cats-car_1	43.9	56.1	50.0
cats-car_2	37.2	45.8	41.5
cats-car_3	83.4	62.8	73.1
cats-car_4	40.1	54.1	47.1
chamaleon_1	84.4	89.2	86.8
deer_1	34.6	37.1	35.9
deer_2	35.6	48.0	41.8
giant-slalom_1	52.8	52.8	52.8
giant-slalom_2	26.0	33.6	29.8
giant-slalom_3	40.3	60.6	50.5
girl-dog_1	81.2	85.6	83.4
girl-dog_2	73.5	77.4	75.5
girl-dog_3	47.9	68.1	58.0
golf_1	58.4	65.1	61.8
golf_2	7.8	36.3	22.1
golf_3	22.4	45.0	33.7
guitar-violin_1	39.1	46.5	42.8
guitar-violin_2	77.3	75.0	76.2
guitar-violin_3	88.9	75.9	82.4
guitar-violin_4	78.6	75.6	77.1

Sequence_Object	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{G}$
man-bike_1	62.2	62.2	62.2
man-bike_2	42.4	44.0	43.2
monkeys-trees_1	40.7	78.7	59.7
monkeys-trees_2	23.1	45.6	34.4
mtb-race_1	54.7	44.7	49.7
mtb-race_2	47.4	48.6	48.0
mtb-race_3	35.5	51.2	43.4
mtb-race_4	74.3	85.5	79.9
orchid_1	80.5	70.7	75.6
orchid_2	75.0	68.9	72.0
people-sunset_1	62.8	60.7	61.8
people-sunset_2	26.5	47.7	37.1
people-sunset_3	17.4	38.5	28.0
people-sunset_4	24.5	35.4	30.0
planes-crossing_1	19.6	20.0	19.8
planes-crossing_2	21.2	36.4	28.8
rollercoaster_1	35.6	46.4	41.0
salsa_1	29.9	50.1	40.0
salsa_2	70.1	74.5	72.3
salsa_3	63.5	73.0	68.3
salsa_4	24.5	52.2	38.4
salsa_5	49.4	63.1	56.3
salsa_6	9.7	32.7	21.2
salsa_7	57.8	62.4	60.1
salsa_8	36.9	52.7	44.8
salsa_9	12.1	31.8	22.0
salsa_10	47.6	60.8	54.2
seasnake_1	15.1	29.5	22.3
skate-jump_1	11.2	38.1	24.7
skate-jump_2	16.0	37.5	26.8
slackline_1	87.0	92.7	89.9

Sequence_Object	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{G}$
gym_1	67.8	64.2	66.0
gym_2	60.2	65.2	62.7
gym_3	30.9	57.8	44.4
gym_4	57.1	85.4	71.3
helicopter_1	86.4	88.3	87.4
helicopter_2	62.1	66.8	64.5
horsejump-stick_1	0.8	16.9	8.9
horsejump-stick_2	69.3	83.9	76.6
horsejump-stick_3	75.7	83.6	79.7
hoverboard_1	81.3	86.6	84.0
hoverboard_2	42.9	66.7	54.8
lock_1	51.4	57.1	54.3
lock_2	79.3	50.6	65.0
lock_3	57.6	63.9	60.8
lock_4	76.7	74.1	75.4

Sequence_Object	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{G}$
subway_1	22.2	33.4	27.8
subway_2	5.2	24.4	14.8
subway_3	47.2	70.6	58.9
subway_4	41.9	56.3	49.1
tandem_1	50.8	80.6	65.7
tandem_2	43.5	57.9	50.7
tandem_3	77.7	67.9	72.8
tandem_4	64.9	71.9	68.4
tennis-vest_1	39.6	66.3	53.0
tennis-vest_2	80.5	94.7	87.6
tractor_1	72.2	64.9	68.6
tractor_2	86.4	59.7	73.1

Table C.1: PiWiVOS' per-object accuracy on DAVIS 2017 test-dev set.

Appendix D. PiWiVOS' visual results on the DAVIS 2017 validation videos

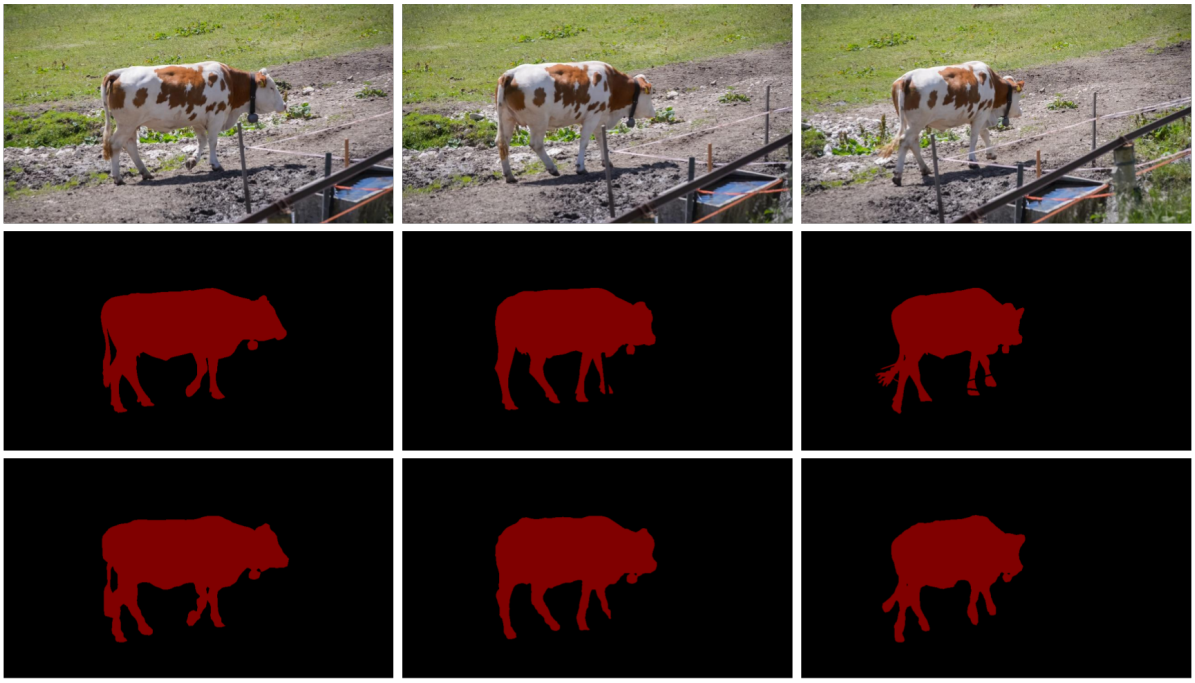


Figure D.1: Qualitative results on the cow sequence.

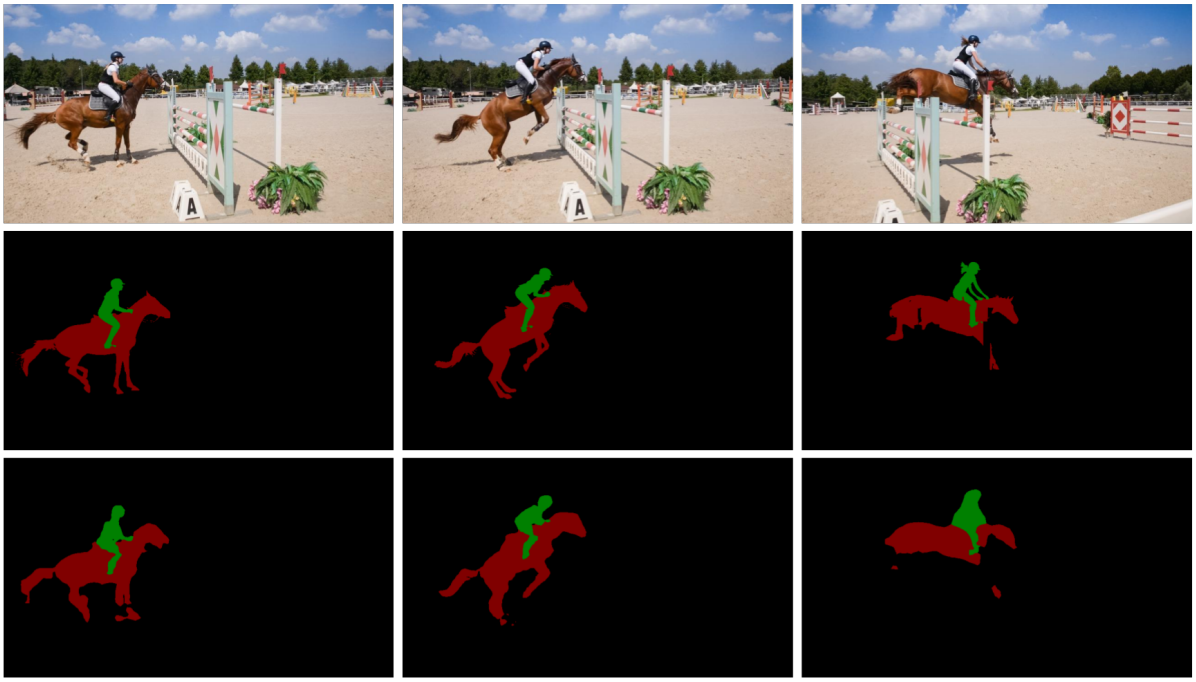


Figure D.2: Qualitative results on the horsejump-high sequence.

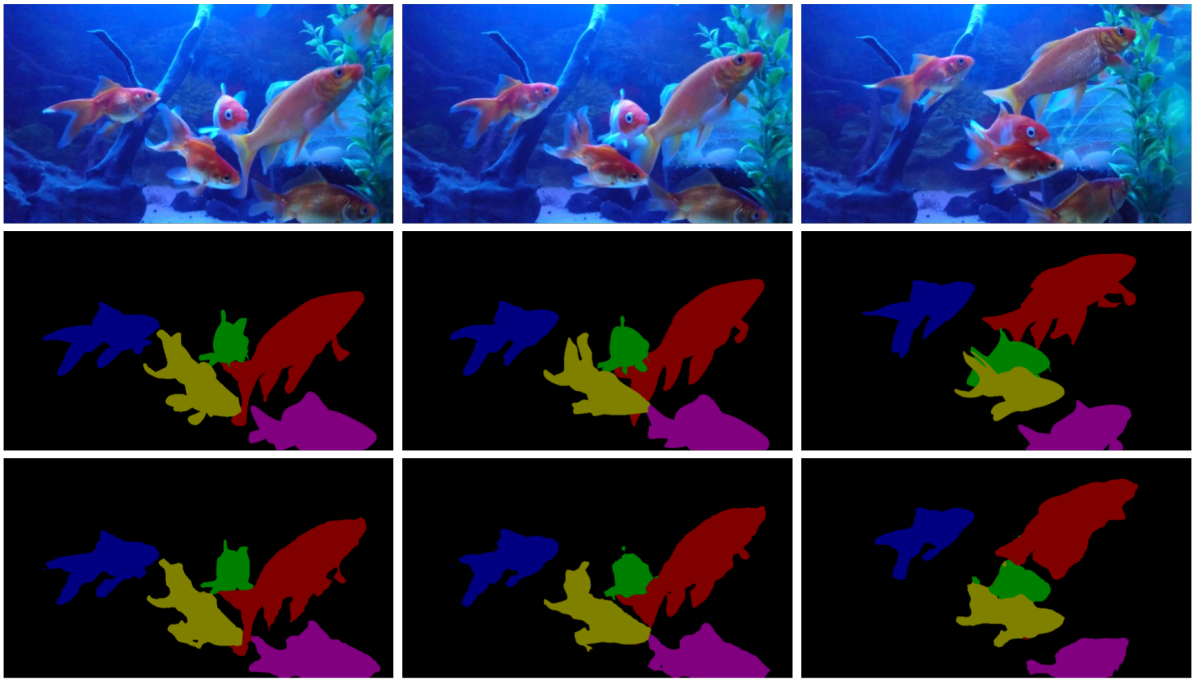


Figure D.3: Qualitative results on the gold-fish sequence.